METHOD FOR CONTROLLING STORAGE DEVICE

Patent Number: JP2001188712
Publication date: 2001-07-10
Inventor(s): KOMATSU SHINPEI; MORITA YUMI; HAYASHI TOMOHIRO; SHIBAZAKI SHOGO; ITO HIROYUKI; TAKEHARA MASARU
Applicant(s): FUJITSU LTD
Requested Patent: JP2001188712
Application Number: JP20000347526 19931001
Priority Number (s):
IPC Classification: G06F12/16; G11C16/02; G11C16/06; G11C29/00
EC Classification:
Equivalents: JP3390740B2

Abstract

PROBLEM TO BE SOLVED: To improve the reliability of a memory in which data cannot be overwritten nor erased by byte units.
SOLUTION: When writing data are transmitted from a main boy processor 2 to a storage device 1, a control means 1b decodes an address by referring to a decoder, and writes data in the new sector or block of a storage area 1a. The decoder is constituted of two stags, and when part of the storage area 1a of the storage device 1 is destroyed, or when part of decoders 1d or 1e is destroyed, one or both of the decoders 1d and 1e in two stages is rewritten. Thus, decoding to the destroyed part is not performed, and even when a storage medium having possibility that part of it is destroyed, such as an EEPROM, a flash memory or the like, is used as the decoder, any probability that the address is turned into an error can be sharply reduced.

(19)日本国特許庁（ＪＰ）　(12)公開特許公報（Ａ）　(11)特許出願公開番号
特開2001－188712
（P2001－188712A）
(43)公開日　平成13年７月10日(2001.7.10)

(51)Int.Cl.⁷		識別記号	FI		テーマコード（参考）
Ｇ０６Ｆ	12/16	３１０	Ｇ０６Ｆ	12/16	３１０Ｑ
					３１０Ｐ
Ｇ１１Ｃ	16/02		Ｇ１１Ｃ	29/00	６０１Ｂ
	16/06				６０１Ｃ
	29/00	６０１			６３１Ｄ

審査請求　有　請求項の数１　ＯＬ　（全 79 頁）　最終頁に続く

(21)出願番号　特願2000－347526(P2000－347526)
(62)分割の表示　特願平5－246547の分割
(22)出願日　平成５年10月１日(1993.10.1)

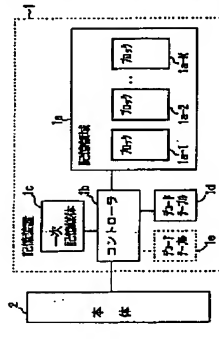
(71)出願人　000005223
富士通株式会社
神奈川県川崎市中原区上小田中４丁目１番１号
(72)発明者　小松　新平
神奈川県横浜市中区本町４丁目36番地　株式会社富士通コンピュータテクノロジ内
(72)発明者　森田　由美
神奈川県横浜市中区本町４丁目36番地　株式会社富士通コンピュータテクノロジ内
(74)代理人　100100930
弁理士　長澤　俊一郎　（外１名）

最終頁に続く

(54)【発明の名称】　記憶装置の制御方法

(57)【要約】
【課題】　上書きができず、また、バイト単位で消去することができないメモリの信頼性を向上させること。
【解決手段】　本体処理装置２から記憶装置１に書き込みデータが送られてくると、制御手段１ｂはデコーダを参照してアドレスをデコードし、データを記憶領域１ａの新たなセクタもしくはブロックに書き込む。デコーダは２段設けられ、記憶装置１の記憶領域１ａの一部が破壊したとき、もしくは、デコータ１ｄ，１ｅの一部が破壊したとき、２段のデコーダ１ｄ，１ｅのいずれか一方、もくしは、両方を書き換える。これにより、破壊した部分へのデコードが行われず、デコーダとして、ＥＥＰＲＯＭ、フラッシュメモリ等のその一部が破壊する可能性のある記憶媒体を用いた場合においても、アドレスがエラーとなる確率を著しく減少させることができる。

本発明の全体の概略構成を示す原理図



【特許請求の範囲】
【請求項１】　読み　きが可能で、かつ、記憶領域の一部が破壊する可能性のある記憶領域(1a)と、記憶領域(1a)中のデータが記憶されている場所を示す書き換え可能なデコーダ(1d)とを備えた記憶装置(1) の制御方法であって、
デコーダを２段設け、記憶装置(1) の記憶領域(1a)の一部が破壊したとき、もしくは、デコータ(1d,1e) の一部が破壊したとき、
２段のデコーダ(1d,1e) のいずれか一方、もくしは、両方を書き換えることにより、破壊した部分へのデコードが行われないようにしたことを特徴とする記憶装置の制御方法。
【発明の詳細な説明】
【０００１】
【産業上の利用分野】本発明はパソコンの外部記憶装置等に使用されるフラッシュ・メモリ等の上書きのできない記憶装置の制御方法に関する。
【０００２】
【従来の技術】近年、パソコン等の外部記憶装置として、フラッシュ・メモリを使った外部記憶装置が注目をあびている。フラッシュ・メモリは、不揮発性のためバックアップ電源が不要であり、電気的に書き換えることができ、しかも安価であるが、次のような問題点を持っている。
①　データを消去してからでなければ、データの書き込みはできず、また、消去の単位はバイト単位は不可能であり、セクタ、ブロックもしくはチップ単位となる。このため、通常のメモリのように、バイト単位でデータを書き換えることはできず、読み出し速度に対して、書き込み速度あるいは消去速度が遅い。
②　消去回数に制限があり、通常、消去回数が１０万～１００万回程度で消去不能となるといわれている。このため、セクタ、もしくはブロックの消去回数が平均化するように使用しないと、消去回数の多い部分が先に不良となり、使用可能領域が減少する。フラッシュ・メモリは上記のような問題点をもっているため、フラッシュ・メモリを使用するにあたっては、データの退避領域を用意し、データの書き換え時にデータを退避したり、管理テーブルを設けてデータの書き込み／消去を管理したり、さらに、不良セクタ、ブロック等が発生した場合に救済措置を講ずるなど、種々の方策が講じる必要がある。
【０００３】
【発明が解決しようとする課題】フラッシュ・メモリは不揮発性で電気的に書き換えることができる等の長所を持っている反面、上記のように多くの問題点を持っており、従来から使用されているＤＲＡＭ、ＳＲＡＭ等の半導体メモリとは異なり、その使用にあたって解決しなければならない問題点も多い。本発明は上記した問題点に鑑みなされたものであって、本発明の目的は、フラッシュ・メモリ等のように上書きができず、また、バイト単位で消去することができないメモリの信頼性を向上させることである。
【０００４】
【課題を解決するための手段】図１ないし図４は本発明の原理図であり、図１は本発明の全体の概略構成を示す原理図、図２～図４は本発明の概要を示す原理図である。図１ないし図４において、１は記憶装置、１ａは上書きすることができず、また、セクタ、ブロック等の所定の単位でしかデータを消去することができない、例えばフラッシュ・メモリ等から構成される記憶領域、１ｂは記憶領域１ａへのデータの書き込み等を制御したり、記憶領域１ａの消去等を行う制御手段、１ｃは記憶領域１ａへのデータの書き込み時に一時的にデータを格納したり、また、記憶領域１ａのデータを退避するための一次記憶媒体、１ｄ，１ｅは記憶領域１ａのアドレスをデコードするためのデコードテーブル、２は本体処理装置である。上記課題を解決するため、本発明の請求項１の発明は、図１，図４に示すように、読み書きが可能で、かつ、記憶領域の一部が破壊する可能性のある記憶領域１ａと、記憶領域１ａ中のデータが記憶されている場所を示す書き換え可能なデコーダ１ｄとを備えた記憶装置において、デコーダを２段設け、記憶装置１の記憶領域１ａの一部が破壊したとき、もしくは、デコータ１ｄ，１ｅの一部が破壊したとき、２段のデコーダ１ｄ，１ｅのいずれか一方、もくしは、両方を書き換えることにより、破壊した部分へのデコードが行われないようにしたものである。
【０００５】
【作用】図１において、本体処理装置２から記憶装置１に書き込みデータが送られてくると、制御手段１ｂはデータを一次記憶媒体１ｃに格納する。そして、デコード・テーブル１ｄ（１ｅ）を参照してアドレスをデコードし、一次記憶媒体１ｃに格納されたデータを記憶領域１ａの新たなセクタもしくはブロックに書き込む。その際、書き込むデータの論理アドレスと同一の論理アドレスのデータが記憶領域１ａに既に存在している場合には、そのデータを消去したり、そのデータを格納した領域に消去可能フラグを立てる。また、データを格納した領域に消去フラグを立てた場合には、制御手段１ｂは所定の時期に所定の単位で、消去不可のデータを一次記憶媒体１ｃに退避して記憶領域１ａを消去し、一次記憶媒体１ｃに退避したデータを記憶領域１ａに書き戻す等の処理を行い、空き領域の作成等を行う。さらに、制御手段１ｂは記憶領域１ａの一部が不良になった場合には、デコード・テーブル１ｄ（１ｅ）を書き換えるなどして、不良部分を予備領域で代替する。
【０００６】図２（ａ）に示す記憶領域１ａが複数のブロック１ａ－１，…，１ａ－Ｎと、該ブロックを区切っ

たセクタから構成され、書き込まれたデータをブロック単位で消去する記憶装置、図４に示す記憶領域１が上書きできない複数のセクタで　成され、セクタに書き込まれたデータをセクタ単位で消去する記憶装置において、データの書き込み／消去は次のように行われる。

(1) 記憶領域１ａが複数のブロック１ａ－１，…，１ａ－Ｎと、該ブロックを区切ったセクタから構成され、書き込まれたデータをブロック単位で消去する記憶装置１において、図２（ｂ）に示すように、ブロック１ａ－１，…，１ａ－Ｎ内の消去不可のデータを空きセクタがある他のブロックに退避するに際して、消去可能なセクタ数が多いブロックからｍブロックと、消去可能が少ないブロックからｎブロック（ｍ，ｎは任意の整数）を選択し、選択されたブロックを同時に空きセクタがある他のブロックに、セクタを散在させて均一に、退避する。これにより、空きブロックを作成することができるとともに、結果的に各ブロックに書き換えが多いセクタと書き換えが少ないセクタが混在することとなり、ブロックの消去回数を平均化することができる。

(2) 記憶領域１ａが、退避領域を含む複数のブロック１ａ－１，…，１ａ－Ｎから構成され、書き込まれたデータをブロック単位で消去する記憶装置において、図２（ｃ）に示すように消去可能なデータを含むブロック１ａ－１，…，１ａ－Ｎのデータの内、消去不可のデータのみを退避領域に移動したのち、上記ブロックのデータを消去し、消去したブロックを新たな退避領域とすることにより、空き領域を作成する。これにより、消去可能データをなくすことができ、本処理を予め行っておくことにより、書き込み時間を短縮することができる。

(3) データの書き込み方向に対して、退避領域の反対側のブロックの消去不可のデータのみを退避領域に移動したのち、上記ブロックのデータを消去し、消去したブロックを新たな退避領域とする。これにより、書き込み場所を示す書き込みポインタと退避領域の間に全ての空き領域を存在させることができる。したがって、処理の中断があっても、書き込みポインタの示す位置からデータを書き込むことができ、処理中断後に必要とする処理量を少なくすることができる。

(4) 記憶領域１が、予備領域を含む複数のブロック１ａ－１，…，１ａ－Ｎから構成され、書き込まれたデータをブロック単位で消去する記憶装置において、図２（ａ）、図３（ａ）に示すように書き込みポインタを予備領域に設定し、データが書き込まれたブロックの消去不可のデータを予備領域に移動したのち、上記ブロックのデータを消去する処理を繰り返し、最後に消去したブロック以外のブロックの空きセクタ数が予備領域にあるセクタ数と等しいか、もしくは、大きくなったとき、予備領域のデータを最後に消去したブロック以外のブロックに移動する。これにより、空き領域を作成することができる。また、空きブロックが使用不可になったとき、使用不可となったブロックを救済して空きブロックを作成することができる。さらに、予備領域のデータを、データを最後に消去したブロック以外のブロックに移動する際、移動先の書き込みポイントを、データを最後に消去したブロックの後のブロックから検索して予備領域のデータを移動するようにすることにより、最後に消去をかけたブロックと書き込みポインタの間に全ての空きセクタを存在させることができ、処理中断後の処理を簡単にすることができる。またさらに、処理中断後、処理を再開したとき、予備領域に消去可能なデータがあるか、あるいは、消去不可のデータのみがあるかに応じて、処理中断時点を判断することができ、処理中断時点の判断を単純化することができる。

(5) 記憶領域１が複数のブロック１ａ－１，…，１ａ－Ｎから構成され、書き込まれたデータをブロック単位で消去する記憶装置１において、ブロック毎に書き込みの有無を示すフラグＦ１を設けて、図３（ｂ）に示すように、フラグＦ１を参照して、書き込まれたデータを消去する。これにより、不要な消去を避けることができ、消去時間の短縮を図ることができるとともに、記憶媒体の寿命を延ばすことができる。また、全記憶領域の消去処理が実行中であることを示す実行中フラグＦ２を設けることにより、全記憶領域の消去処理を実行中に、処理が中断された際、実行中フラグを参照して、全記憶領域の消去処理中であることを判断することができ、不意の処理中断に対応することができる。

(6) 記憶領域１ａ上に同一の機能に対応した複数ビットのフラグを記録し、該フラグの論理積によりフラグの判定を行うようにする。これにより、全ビットが不良にならない限り、正しい判定値を常に出力することができ、システムの信頼性を向上させることができる。

(7) 図４に示すように、記憶領域１が上書きできない複数のセクタで構成され、セクタに書き込まれたデータをセクタ単位で消去する記憶装置において、同一のセクタ番号を持つセクタを少なくとも２個用意し、同一番号が付されたセクタのいずれか一方にデータがあった場合、他方のセクタにデータを書き込むと同時に、一方のセクタのデータを消去する。これにより、書き込み速度を向上することができる。

(8) 図４に示す記憶装置において、書き込みするセクタにデータがある場合、データを一次記憶装置１ｃに転送している間に、上記セクタのデータを消去し、データの消去完了後、もしくは、データ転送の終了後、上記セクタにデータを書き込む。これにより書き込み速度を向上することができる。

(9) 図４に示す記憶装置において、不良なセクタを代替する予備の領域１ｂ－１，１ｂ－２と、書き込みセクタを管理する書き換え可能なデコード・テーブル１ｄを設け、不良なセクタが発生したとき、上記デコード・テーブル１ｄを書き換えることにより、予備の領域１ｂ－

１，１ｂ－２に書き込みセクタを変更し、予備の領域１ｂ－１，１ｂ－２がなくなつたとき、デコード・テーブル１ｄを書き換えてセクタを再構成するようにしたので、不良セクタ発生時の救済を行うことができる。

(10)図４に示す記憶装置において、Ｎ個のセクタ番号に対してＮ＋１個のセクタを設けるとともに、Ｎ＋１個の各セクタをＮ個のセクタ番号で共有し、データをセクタへ書き込む際、データをＮ＋１個のセクタの内の空いているセクタに書き込む。これにより、書き込み速度を向上することができるとともに、余分に用意するセクタの数を少なくすることができ、コストダウンを図ることができる。また、不良なセクタを代替する予備の領域１ｂ－１，１ｂ－２を設け、不良なセクタが発生したとき、デコード・テーブル１ｄを書き換え、予備の領域１ｂ－１，１ｂ－２に書き込みブロックあるいはセクタを変更するようにしたり、デコード・テーブル１ｄを再編成する。これにより、不良セクタ発生時の救済を行うことができる。

(11)記憶領域１ａが複数に分割されており、分割されたチップ内に複数のブロックが設けられ、チップ内のブロックにデータを書き込む際、新たなブロックにデータを書き込み、書き込んだデータがチップ内に既に存在している場合には、そのデータをブロック単位で消去する記憶装置において、データを書き込むチップを、書き込むデータに対応させて固定する。これにより、制御手段１ｂは各チップ内のデータのみを管理すればよく、全領域を管理する必要がない。このため、制御手段１ｂにおける管理テーブルを簡単にすることができ、書き込み速度を向上させることができる。また、各チップ内にデータを書き込むための複数のワークブロックを設けることにより、ワークブロックに不良が発生した場合に、不良となったワークブロックを救済することができる。

【０００７】本発明の請求項１の発明は、上述の読み書きが可能で、かつ、記憶領域の一部が破壊する可能性のある記憶領域１ａと、記憶領域１ａ中のデータが記憶されている場所を示す書き換え可能なデコーダ１ｄとを備えた記憶装置において、図４に示すようにデコーダを２段設け、記憶装置１の記憶領域１ａの一部が破壊したとき、もしくは、デコータ１ｄ，１ｅの一部が破壊したとき、２段のデコーダ１ｄ，１ｅのいずれか一方、もくしは、両方を書き換えることにより、破壊した部分へのデコードが行われないようにしたので、デコーダとして、ＥＥＰＲＯＭ、フラッシュメモリ等のその一部が破壊する可能性のある記憶媒体を用いた場合においても、アドレスがエラーとなる確率を著しく減少させることができ、信頼性を確保することができる。

【０００８】

【実施例】次に、本発明の実施例を説明する。

Ａ．本発明の前提となる記憶装置の構成。

図５は本発明の前提となるフラッシュ・メモリを使用した記憶装置の構成を示すブロック図である。同図において、２０は例えばメモリ・カード等の記憶装置、２１はフラッシュ・メモリのデータの書換え、消去等を制御するコントローラＬＳＩ、２２はプロセッサ、２３は各種テーブルを格納したり、データの書き込み時のデータ・バッファあるいは退避バッファとして使用されるＳＲＡＭ、２４はクロック発振器、２５－１～２５－５はフラッシュ・メモリである。同図において、本体よりＳＲＡＭ２３に転送されたデータをフラッシュ・メモリ２５－１～２５－５に書き込む際、前記したようにフラッシュ・メモリは上書きができないので、データを書き込み場合には、新たな領域にデータを書き込む。また、データ更新の場合には、旧データに消去フラグを立てる等の措置を講ずる。そして、旧データに消去フラグが立てられている場合には、所定の時期に、消去フラグが付されていないデータのみを所定の単位でＳＲＡＭ２３の退避領域に退避し、フラッシュ・メモリの書き込むべき領域を消去したのち、データを書き戻すことによりデータの整理を行う。

【０００９】図６は上記したＳＲＡＭ２３の内容を示す図であり、同図に示すように、ＳＲＭ２３には下記のデータが格納される。

(1) 消去回数テーブル

フラッシュ・メモリの各ブロックの消去回数を保持している。

(2) 消去可能セクタ数テーブル

各ブロックの消去可能セクタ・フラグの立っている総数値を保持する。

(3) 書き込みポインタ

フラッシュ・メモリに書き込みを開始するチップNo.，ブロックNo.，セクタ・アドレスNo.を保持する。

(4) ＷＯＲＫ－ＢＬＯＣＫ－No.

現在のＷＯＲＫ－ＢＬＯＣＫ－No.を示し、チップNo.，ブロックNo.を保持する。

(5) 整理ポインタ

現在、整理をしているチップNo.，ブロックNo.，セクタ・アドレスNo.を保持する。

(6) 書き換え有無

現在、書き込みするデータが新規であるかどうかを示す。

(7) 書き込み回数

整理時の本体書き込み回数を管理する。

(8) 退避カウンタ

整理時の退避動作のセクタ数を管理する。

(9) チップ数

カードの搭載フラッシューチップ数を保持する。

(10)セクタマップ・テーブル

論理アドレスの変換用にチップNo.，ブロックNo.，セクタ・アドレスNo.を保持する。

【００１０】図７、図８はフラッシュ・メモリ２５－１

～２５－５の内容を示す図であり、フラッシュ・メモリの各セクタには、次の(1) から(5) に示す制御情報、データ等が格納されている。なお、この例においては、セクタが１２６設けられており、各セクタには、(1) から(5) の管理情報等とデータが格納され、１２６番目のセクタの後に、次の(6) から(14)の管理情報等が格納されている。

(1) 不良フラグ
このセクタの状態を示し、使用不可の場合に、ここに書き込みを行う。
(2) 消去フラグ
このセクタのデータの状態を示し、書き換え等で無効になった場合に、ここに書き込みを行う。
(3) 論理アドレス
このセクタの論理アドレスを示す。
(4) データ
このセクタに書き込まれたデータ。
(5) チェックサム
書き込んだデータのチェックサム
(6) 不良セクタ・メモリ
整理対象ブロックの中の不良セクタを示す。
(7) 整理元消去回数
整理対象ブロックの消去回数。
(8) 退避ブロックNo.
データを退避してくるチップNo. , ブロックNo.
(9) 消去Ｓｔａｒｔ
整理対象ブロックの消去を開始する時に立てる。
(10)消去Ｅｎｄ
整理対象ブロックの消去を終了するときに立てる。
(11)Ａｌｌ　Ｅｒａｓｅ対象
Ａｌｌ　Ｅｒａｓｅ時に消去対象として確定した時に立てる。
(12)空きブロック
ブロック内にデータがあるかを示し、データがある場合に立てられる。
(13)ブロック・ステータス
このブロックの状態を示し、使用不可になった場合に立てる。

【００１１】図９ないし図１３はフラッシュ・メモリ２５－１から２５－５へのデータ書き込み時のプロセッサ２２における処理を示すフローチャートであり、同図を参照して書き込み時の動作を詳細に説明する。図９のステップＳ１において、ＳＲＡＭ２３の、現在書き込みをするデータが新規であるか否かを示す「書き換えの有無」（図６参照）を０として、ステップＳ２において、整理中であるか否かを判別する。整理中でない場合には、ステップＳ５に行き、ＳＲＡＭ２３の、整理時の本体書き込み回数を管理する「書き込み回数」を０にしステップＳ７に行く。また、整理中の場合には、ステップＳ３に行き、「書き込み回数」に１を加え、ステップＳ４において、書き込み回数が６であるか否かを判別する。　き込み回数が６の場合には、ステップＳ６に行き、書き込み回数を１にする。また、書き込み回数が６でない場合には、ステップＳ７に行く。上記処理は、後述するように、書き込み回数が２～５の場合には整理動作を行わないようし、書き込み回数が１の時のみ整理動作を行うことにより、頻繁な整理動作を避けるためであり、上記のように書き込み回数を設定することにより、５回に１回整理動作を行うこととなる。ステップＳ７において、書き込むデータの論理アドレスがオーバーしているか否かを判別し、オーバーしている場合には、エラー処理を行う。論理アドレスがオーバーしていない場合には、ステップＳ８に行き、旧データかあるか否か、すなわち、現在書き込むデータが新規なデータであるか否か判別し、新規なデータでない場合には、ステップＳ９において、ＳＲＡＭ２３の「書き換えの有無」を１にしてステップＳ１１に行く。また、新規なデータの場合には、ステップＳ１０において、ＳＲＡＭ２３の「書き換えの有無」を０にしてステップＳ１１に行く。ついで、ステップＳ１１において、本体からデータをＳＲＡＭに転送し、ステップＳ１２において転送エラーがあるか否かを判別する。転送エラーがある場合にはエラー処理を行う。転送エラーがない場合には、図１０のステップＳ１３に行き、書き換えの有無、すなわち、新規データであるかどうか判別し、新規データでない場合には、データの更新であるので、ステップＳ１４において旧データのあるセクタに消去ビットを書き込み、ステップＳ１５に行く。

【００１２】ステップＳ１５において、ＳＲＡＭ２３の整理時の書き込み回数を管理する「書き込み回数」が１であるか否かを判別し、「書き込み回数」が１の場合には、ステップＳ１６以降でフラッシュ・メモリの整理を行う。また、「書き込み回数」が１でない場合には、図１３のステップＳ４６に行き、ステップＳ４６以降で、ＳＲＡＭ２３のデータをフラッシュ・メモリへ書き込む。ステップＳ１６において、ＳＲＡＭ２３上の整理時における退避動作のセクタ数を管理する「退避カウンタ」を０にして、ステップＳ１７において、現在整理をしているセクタアドレスNo. を示す「整理ポインタ」（セクタアドレス）が１２６セクタより多いか否かを判別し、１２６より小さくない場合には、整理が終了したものとして、図１２のステップＳ３８に行く。また、「整理ポインタ」（セクタアドレス）が１２６セクタより少ない場合には、ステップＳ１８に行き、整理ポインタが０であるか否かを判別し、整理ポインタが０でない場合にはステップＳ２２に行く。

【００１３】整理ポインタが０の場合には、ステップＳ１９において、各ブロックの消去回数や消去可能セクタ数をＳＲＡＭ２３のテーブルから求め、それをもとにして、整理対象を選択する。ついで、ステップＳ２０にお

いて、フラッシュ・メモリ上のWORK－BLOCKの退避ブロックNo.（図8参照）に整理対象のチップNo.，ブロックNo.を書き込む。ステップS21において、書き込みエラーがあるか否か判別し、書き込みエラーがある場合には、エラー処理を行い、書き込みエラーがない場合には、ステップS22に行く。ステップS22において、退避するデータを検索する。すなわち、フラッシュ・メモリのデータに消去フラグ（図7参照）が書き込まれている場合には、次のセクタに行く。また、論理アドレスが無い場合には消去フラグを書き込み、次のセクタに行き、論理アドレスが異常のものは消去フラグと不良フラグを書き込み、次のセクタに行く。

【0014】次いで、図11のステップS23に行き、書き込みエラーがあるか否かを判別し、書き込みエラーがない場合には、ステップS24において、整理ポインタ（セクタアドレス）が126番目のセクタを指しているか否かを判別し、整理ポインタが126である場合には、整理が終わったものとして、図12のステップS38に行く。整理ポインタが126でない場合には、ステップS25に行き、退避するデータをフラッシュ・メモリからSRAM23に移動し、ステップS26において、生成したチェックサムがフラッシュ・メモリ上のチェックサム（図7参照）と一致するか否かを判別する。一致しない場合には、ステップS27において、チェックサムがFFhであるか否かを判別する。そして、チェックサムがFFhでない場合には、ステップS28にいき、フラッシュ・メモリ上のチェックサムをFFhとし、データ退避元のセクタの不良フラグを書き込み、ステップS29に行く。ステップS29において、書き込みエラーが有るか否か判別し、ステップS30に行く。また、ステップS26において、チェックサムが一致するか、S27において、チェックサムがFFhの場合には、ステップS30に行き、書き込みポインタが示すセクタから書き込めるセクタを探し、ステップS31において、書き込み可能セクタがあるか否かを判別する。書き込み可能セクタがない場合には、エラー処理を行い、書き込み可能セクタがある場合には、ステップS32に行き、SRAM23上のデータをフラッシュ・メモリに移動する。

【0015】ステップS33において、書き込みエラーが有るか否かを判別し、書き込みエラーがある場合には、ステップS30に戻る。書き込みエラーがない場合には、図12のステップS34に行き、データ退避元のセクタの消去フラグを書き込む。そして、ステップS35において、書き込みエラーがあるか否かを判別し、書き込みエラーがない場合には、ステップS36において、SRAM23上のセクタマップテーブルを書き換えるとともに、退避カウンタに1を加える。ついで、ステップS38において、退避動作が予め定められた所定回数（この場合には60回）行われたか否かを判別し、退避動作が60回になった場合には、ひとまずフラッシュメモリの整理を終えてステップS38に行く。また、退避動作が60回になっていない場合には、図10のステップS22に戻り上記処理を繰り返す。

【0016】退避カウンタが60になっているか、前記した図10のステップS17において、整理ポインタが126より小さくないと判別された場合には、ステップS38に行き、退避カウンタが0であるか否か判別する。そして、退避カウンタが0である場合、つまり、整理ポインタが126より小さくない場合であって、かつ、退避動作が行われていない場合には、ステップS39に行き、ステップS39以降で、不良フラグの情報を不良セクタメモリに書き込む等の処理を行う。すなわち、退避カウンタが0である場合には、処理時間が短かかったので、S39以降の不良フラグの情報を不良セクタメモリに書き込む等の処理を行い、また、退避カウンタが0でない場合には、図13のステップS46に行き、SRAM23上のデータをフラッシュ・メモリに移動する等の処理を行う。

【0017】ステップS39において、退避したブロックの不良フラグの情報を書き込み先（書き込み先はWORK－BLOCK）の不良セクタ・メモリ（図8参照）に書き込み、また、消去回数を整理元消去回数欄（図8参照）に書き込む。ステップS40において、書き込みエラーがあるか否かを判別し、書き込みエラーがない場合には、ステップS41に行き、整理したブロックを消去する。ステップS42において、消去エラーがあるか否か判別し、消去エラーがない場合には、図13のステップS43に行き、消去したブロックに不良セクタメモリから不良フラグ情報を戻し、整理元消去回数に1を加えて、消去回数に書き込む。ついで、ステップS44において、消去したブロックに対応するテーブルの消去回数に1を加え、消去可能セクタ数を0にする。ステップS45において、整理ポインタをWORK－BLOCKに入れ、整理ポインタを0とする。

【0018】次にステップS46に行き、書き込みポインタの示すセクタから書き込めるセクタを探し、ステップS47において書き込み可能セクタが有るか否かを判別する。書き込み可能セクタがある場合には、ステップS48に行き、SRAM23上のデータをフラッシュ・メモリに移動し、ステップS49において、書き込みエラーがあるか否かを判別する。書き込みエラーが有る場合には、ステップS46に戻り、書き込みエラーがない場合には、ステップS50に行く。ステップS50において、書き換え有無が1であるか否か、すなわち、書き込みするデータが新規であるか否か判別し、書き換えの有無が1の場合には、ステップS51で消去フラグを書き込んだセクタに対応する消去可能セクタ数テーブル（図6参照）の値に1を加え、ステップS52に行く。ステップS50において、書き換え有無が0である場合

には、ステップ５２において、セクタマップテーブル（図６参照）を書き換え、ステップＳ５３において、次の書き込みに備えて、書き込みポインタの示すセクタから書き込めるセクタを探しておき終了する。

【００１９】Ｂ．書き込み／消去処理。

前記したように、フラッシュ・メモリは、消去回数に制限があり、全てのメモリを有効に活用するためには、消去回数を平均化する必要がある。また、フラッシュ・メモリは上書きをすることができず、データを書き換える場合には、新たなセクタにデータを書き込んで、旧データに消去可能フラグを立てておくなどの措置を講じ、適宜の時点で消去可能フラグが立っているセクタを消去する必要がある。このため、消去可能なデータを消去してデータを書き込むための空き領域を確保する必要があるとともに、上記空き領域が、不良ブロックが発生するなどの原因により使えなくなったとき、救済措置を講じて新たな空き領域を作成する必要がある。以下に、上記した消去回数の平均化（実施例１）、消去可能なデータを消去して空き領域を確保する空き領域の作成（実施例２）、不良ブロック発生時の空き領域作成のための救済措置（実施例３）等の領域処理についての本発明の実施例を示す。

【００２０】（１）実施例１（消去回数の平均化）

前記したように、フラッシュ・メモリは、消去回数に制限があり、全てのメモリを有効に活用するためには、消去回数を平均化する必要がある。本実施例は、各ブロックの消去回数を意識することなく、消去回数のバラツキを抑えることが可能な実施例を示しており、本実施例においては、セクタ単位で追記書き込み、ブロック単位で消去するメモリにおいて、ブロックをセクタ単位で区切り、以下に示すように、消化可能なセクタが多いブロックと、消去可能なセクタが少ないブロックとを混ぜ合わせて新しいブロックに書き込むことにより、消去回数を平均化している。次に本実施例について説明する。図１４ないし図１８は本実施例における書き込み処理を示す図であり、本実施例においては、６セクタを持つ６ブロックのフラッシュ・メモリにＡからＯの論理セクタを書き込む例を示している。また、以下の説明においては、ブロック内の空きセクタの数がＮ個（本実施例では２個）になった場合にＭ個のブロック（本実施例では２個）に退避動作を行うこととし、また、消去可能セクタ数が多い方からｍブロック（本実施例では１ブロック）と消去可能セクタの少ない方からｎブロック（本実施例では１ブロック）を選択して、同時に退避する。なお、退避動作を行う場合には、前記したように、消去可能フラグが立っていないセクタを読み出して、図５に示すＳＲＡＭ２３に移動し、ついで、ＳＲＡＭ２３に移動したセクタを新たなブロックに書き込む。

【００２１】図１４の（ａ）において、同一のセクタがないので、ブロック１とブロック２にセクタＡ，Ｂ，Ｃ，Ｄ，Ｅ，Ｆ，Ｇ，Ｈ，Ｉを、最初のセクタから順番に書き込む。（ｂ）において、セクタＣ，Ｄを　き込む。この場合には、同一のセクタＣ，Ｄがあるので、既にＣ，Ｄが書き込まれているセクタに消去可能フラグを立て、新しいセクタＣ，Ｄを書き込む。ついで、図１５の（ｃ）において、セクタＪ，Ｋ，Ｌ，Ｍ，Ｎ，Ｏを書き込む。この場合にも、（ａ）の場合と同様、同一のセクタがないので、既に書き込まれたセクタの後に順番に書き込む。（ｄ）において、セクタＨ，Ｉ，Ｊ，Ｋ，Ｍ，Ｎを書き込む。同一セクタが書き込まれているので、消去可能フラグを立て、新しいセクタＨ，Ｉ，Ｊ，Ｋ，Ｍ，Ｎを書き込む。

【００２２】図１６（ｅ－１），（ｅ－２）において、セクタＣ，Ｄを書き込む。その際、空きブロックの数が２個になっているので、退避動作を行う。すなわち、図１５の（ｄ）において、消去可能なセクタが最も多いブロック３と、消去可能セクタが最も少ないブロック４を選択し、ブロック３のセクタとブロック４のセクタを混在させ、退避先の各ブロックのセクタ数が均一になるように、空きブロック５と６に書き込む。その結果、図１６（ｅ－１）に示すように、セクタＯ，Ｉ，Ｋ，Ｍがブロック５に、また、セクタＨ，Ｊ，Ｌ，Ｎがブロック６に書き込まれ、ブロック３と４が空きブロックとなる。この状態で、既にＣ，Ｄが書き込まれたブロック２に消去可能フラグを立て、セクタＣ，Ｄを図１６（ｅ－２）に示すようにブロック５に書き込む。

【００２３】図１７（ｆ－１）において、セクタＨ，Ｉ，Ｊを書き込む。この場合にも、同一のセクタが既にあるので、ブロック５とブロック６に消去可能フラグを立て、ブロック６にセクタＨ，Ｉを書き込むが、セクタＪを書き込む時点で空きブロックが２個になったので、前記と同様退避動作を行う。すなわち、（ｆ－２）に示すように、消去可能なセクタが最も多いブロック２と、消去可能セクタが最も少ないブロック６を選択し、ブロック２のセクタとブロック６のセクタを混在させ、退避先の各ブロックのセクタ数が均一になるように、空きなブロック３と４に書き込む。ついで、（ｆ－３）に示すように、セクタＪを書き込む。この場合にも同一セクタがあるので、既に書き込まれているブロック４のセクタに消去可能フラグを立て、ブロック３にセクタＪを書き込む。

【００２４】図１８（ｇ）において、セクタＥ，Ｆ，Ｇを書き込む。この場合にも同一セクタがあるので、既に書き込まれているブロック１，３のセクタに消去可能フラグを立て、ブロック３と４にセクタＥ，Ｆ，Ｇを書き込む。図１９は本実施例の処理を示すフローチャートであり、同図により本実施例の処理について説明する。ステップＳ１において、本体よりデータを受け取ると、ステップＳ２のおいて、空きセクタ有りのブロック数がＮ以下であるか否か判別する。空きセクタ有りのブロック

数がN以下でない場合には、ステップS7に行く。空きセクタ有りのブロック数がN以下の場合には、ステップS3において、消去可能セクタ数が多いブロックからmブロックを退避対象とし、ステップS4において、消去可能セクタ数が少ないブロックからnブロックを退避対象とする。ステップS5において、退避対象のブロックからデータを空きブロックに移動する。その際の書き込み方は、データを移動する毎に書き込みブロックを変え、M個のブロックに書き込んだら最初に書き込んだブロックに戻る。ついで、ステップS6において、退避対象のブロックを消去する。ステップS7において、同一論理セクタがあれば、既に書き込まれている物理セクタに消去可能フラグを立て、ステップS8において、本体から受け取ったデータをフラッシュ・メモリに書き込む。

【0025】ところで、一般にセクタには、システム・プログラムのように殆ど書き換えが行われないデータを格納したセクタと、常時書き換えが行われるデータを格納したセクタがあり、普通にデータの書き込み/退避をおこなっていると、殆ど書き換えが行われないセクタが多いブロックは消去回数が少なくなる。したがって、上記のように、消去可能なセクタが多いブロック（常時書き換えが行われるセクタが多いブロック）と、消去可能なセクタが少ないブロック（殆ど書き換えが行われないセクタが多いブロック）とを混ぜ合わせて新しいブロックに書き込むことにより、結果的に各ブロックに書き換えが多いセクタと書き換えが少ないセクタが混在することとなり、ブロックの消去回数を平均化することができる。本実施例においては、上記のような原理に基づき書き込み処理をおこなっているので、各ブロックの消去回数を意識することなく、消去回数のバラツキを抑えることができ、フラッシュ・メモリのように消去回数が有限なメモリの寿命を延ばすことが可能となる。また、消去回数をカウントすることなく消去回数のバラツキを抑えることができるので、消去回数をカウントするための領域を設けることなく、メモリの管理を行うことも可能となる。

【0026】（2）実施例2（消去可能データの消去による空き領域の作成）

前記したように、フラッシュ・メモリは上書きをすることができず、消去した後でなければ書き込みを行うことができない。このため、書き込み時、同一のセクタがある場合には消去可能フラグを立て、適宜の時点で消去可能フラグが立っているセクタを消去する必要がある。本実施例は、上記のように、消去フラグがたっている消去可能なデータを記憶媒体上からなくすための領域処理を示しており、本実施例による領域処理を処理の空き時間などに予め行っておくことにより、書き込みできる領域を確保でき、書き込み時間を短縮することができる。

【0027】図20から図23は本実施例における書き込み/消去処理を示す図であり、同図により本実施例を説明する。なお、本実施例においては、次の点を前提としている。

① ブロック数が6個で、1ブロックあたり、6個のセクタがあり、リード/ライトはセクタ単位で行い、消去はブロック単位で行う。

② 書き込み方式は追記書き込み方式であり、同じアドレスの論理セクタを書き込むときは、旧論理セクタに格納されている物理セクタに消去可能フラグを立てる。

③ 追記のために書き込み領域を5ブロック+退避領域1ブロック（ブロック1～ブロック6）とする。

④ 書き込み領域がなくなった場合には、消去可能フラグが立っている物理セクタが一番多いブロックの消去不可の物理セクタのデータを退避領域に移動して、移動元のブロックを消去する。そして、そのブロックを退避領域として使用し、今までの退避領域を書き込み領域とする（この一連の動作を退避動作と呼ぶ）。

⑤ 本体から送られて来るセクタのアドレスはAからPとする。

⑥ ブロック中に消去可能なデータがm以上あるとき、本実施例における領域処理の対象とする（図20から図23の例においては、m=1としている）。なお、図20から図23中のブロックの右に付された矢印は書き込み場所を示す書き込みポインタの位置である。

【0028】図20（a）において、論理セクタA～Gを書き込む。この場合には、同一アドレスの論理セクタがないので、書き込みポインタが指す位置から順番に書き込む。（b）において、論理セクタA，D～Gを書き込む。この場合には、同アドレスの論理セクタ（A，D，E，F，G）があるので、消去可能フラグを立てた後に、（a）と同様に書き込む。（c）において、論理セクタH～Nを書き込む。この場合には、同一アドレスの論理セクタがないので、そのまま書き込む。図21（d）において、A，G～Lを書き込む。この場合には、同アドレスの論理セクタ（A，G，H，I，J，K，L）があるので、消去可能フラグを立てた後に書き込む。（e）において、A，H～Jを書き込む。この場合にも、同アドレスの論理セクタ（A，H，I，J，）があるので、消去可能フラグを立てた後に書き込む。（f-1）において、A，I～Lの書き込みを行おうとするが書き込み領域がないので、退避動作をを行う。すなわち、消去可能フラグが立っている物理セクタが一番多いブロック（ブロック3）のデータを退避領域（ブロック6）に移動して、移動元のブロックを消去する。そして、そのブロック（ブロック3）を退避領域として使用し、今までの退避領域（ブロック6）を書き込み領域とする。その結果、（f-1）のようになる。

【0029】ついで、図22（f-2）に示すように、A，I～Lを書き込む。この場合には、同アドレスの論理セクタがあるので、消去可能フラグを立てた後にA，

Ｉ～Ｌを書き込む。この時、書き込みポインタは退避領域（ブロック６）の先頭にある。以上のよう書き込むことにより、消去可能なデータを増えたので、本実施例による領域処理を行う。すなわち、退避領域（ブロック３）から見て、書き込み方向に対して一つ反対方向のブロック（ブロック２）を処理対象として、ブロック２と３で退避動作を行う。その結果、図２２（ｇ－１）に示すように、ブロック２が退避領域となり、ブロック３にブロック２の消去不可のデータが移される。次に上記と同様に、ブロック１とブロック２で退避動作を行う。その結果（ｇ－２）に示すように、ブロック１が退避領域となり、ブロック２にブロック１の消去不可のデータが移される。

【００３０】ついで、上記と同様にブロック５と１で退避動作を行う。その結果、図２３（ｇ－３）に示すように、ブロック５が退避領域となり、ブロック１にブロック５の消去不可のデータが移される。なお、ブロック６には消去可能のデータがないので、領域処理の対象としない。同様にブロック４と５で退避動作を行う。その結果、（ｇ－４）に示すように、ブロック４が退避領域となり、ブロック５にブロック４の消去不可のデータが移される。そして、退避領域の位置が処理を行う前の退避領域の位置まできたので、領域処理を終わる。以上のような処理を行うことにより、消去可能データは記憶媒体上からなくなる。次に（ｈ）に示すように、Ａ，Ｈ～Ｊを書き込む。この場合にも、同アドレスの論理セクタがあるので、消去可能フラグを立てたのち書き込む。

【００３１】図２４は本実施例の処理を示すフローチャートであり、同図により本実施例の処理を説明する。ステップＳ１において、現在の退避領域のブロックNo.を記憶し、ステップＳ２において、退避領域を処理対象とする。ステップＳ３において、処理対象を書き込み方向とは反対方向に一つ進める。ステップＳ４において、処理対象とステップＳ１において保存した保存ブロックNo.を比較し、同じであれば処理を終了する。処理対象と保存ブロックNo.が異なる場合には、ステップＳ５に行き、処理対象の消去可能セクタ数を所定値ｍと比較し、消去可能セクタ数がｍ以下の場合には、ステップＳ３に戻り、上記処理を繰り返す。消去可能セクタ数がｍより大きい場合には、ステップＳ６に行き、処理対象を退避対象として、ステップＳ７において、書き込みポインタを退避領域の先頭に移動する。ステップＳ８において、退避対象のブロックからデータを移動する。ステップＳ９において、退避対象のブロックを消去し、ステップＳ１０において、退避対象のブロックを退避領域とし、ステップＳ３に戻り上記処理を繰り返す。

【００３２】本実施例においては、上記のようにして領域処理を行っているので消去可能データを無くすことができ、本実施例の領域処理を予めおこなっておくことにより、書き込み時間の短縮を図ることができる。また、領域処理の対象となるブロックを書き込み方向とは反対方向に進めているので、書き込み場所を示している書き込みポインタと退避領域の間に全ての空き領域を存在させることができる。その結果、どのような場合に処理の中断がおこっても、書き込みポインタの示す位置からデータを書き込んで行くことができ、処理中断後に必要とする処理を少なくすることができ、また、処理を全ておえていなくても、支障がない。

【００３３】（３）実施例３（空き領域の作成のための救済措置）

前記したように、フラッシュ・メモリはデータ書込みのため、メモリ中に空き領域を確保しておく必要があるが、本実施例は、上記空き領域が、不良ブロックが発生するなどの原因により使えなくなったとき、救済措置を講じて新たな空き領域を作成する実施例を示している。図２５から図３１は本実施例の空き領域の作成処理を示す図であり、同図により本実施例を説明する。なお、本実施例においては次の点を前提としている。

① ブロック数が７個で、１ブロックあたり、６個のセクタがあり、リード／ライトはセクタ単位で行い、消去はブロック単位で行う。

② 書き込み方式は追記書き込み方式であり、同じアドレスの論理セクタを書き込むときは、旧論理セクタに格納されている物理セクタに消去可能フラグを立てる。

③ 障害時の救済用に１ブロックの予備領域（ブロック０）を確保し、追記のために書き込み領域を５ブロック＋退避領域１ブロック（ブロック１～ブロック６）とする。

④ 書き込み領域がなくなった場合には、消去可能フラグが立っている物理セクタが一番多いブロックの消去不可の物理セクタのデータを退避領域に移動して、移動元のブロックを消去する。そして、そのブロックを退避領域として使用し、今までの退避領域を書き込み領域とする（この一連の動作を退避動作と呼ぶ）。

⑤ 本体から送られて来るセクタのアドレスはＡからＰとする。

【００３４】図２５（ａ）において、セクタＡ～Ｇを書き込む。この場合には、同じ論理セクタがないので、そのまま、ブロック１から書き込む。（ｂ）において、セクタＡ，Ｄ～Ｇを書き込む。この場合には、セクタＡ，Ｄ～Ｇと同一論理セクタがあるので、同一の論理セクタに消去可能フラグを立てたのち書き込む。（ｃ）において、セクタＨ～Ｎを書き込む。この場合には、同じ論理セクタがないので、そのまま書き込む。図２６（ｄ）において、セクタＡ，Ｇ～Ｌを書き込む。この場合、セクタＡ，Ｇ～Ｌと同一論理セクタがあるので、同一の論理セクタに消去可能フラグを立てたのち書き込む。（ｅ）において、セクタＡ，Ｈ～Ｊを書き込む。この場合、セクタＡ，Ｈ～Ｊと同一論理セクタがあるので、同一の論理セクタに消去可能フラグを立てたのち書き込む。

【００３５】図２７（ｆ－１）において、セクタＡ，Ｉ～Ｌを書き込むが、この場合には、書き込み領域がないので、退避動作を行った後書き込み処理を行う。すなわち、消去可能フラグが立っているデータが最も多いブロック（ブロック３）の消去可能でないデータを退避領域（ブロック６）に移動して、移動元のブロックを消去する。そして、そのブロック（ブロック３）を退避領域として使用し、今までの退避領域を書き込み領域とする。ついで、（ｆ－２）において、セクタＡ，Ｉ～Ｌを書き込み領域となったブロック６に書き込む。次に、（ｇ）において、Ａ，Ｍ，Ｎを書き込みを行うが、上記（ｆ）と同様書き込み領域がないので、まず、退避動作を行う。すなわち、消去可能フラグが立っているデータが最も多いブロック（ブロック５）の消去可能でないデータを退避領域（ブロック３）に移動して、移動元のブロックを消去する。そして、そのブロック（ブロック５）を退避領域として使用し、今までの退避領域を書き込み領域とする。ここで、ブロック５が消去失敗により使用不可となったとすると、退避領域がなくなるので救済措置を行う。

【００３６】そこで、図２８（ｈ）に示すように、予備ブロック０を仮の退避領域として退避動作を行う。すなわち、図２７（ｇ）において、消去可能でないデータが最も多いブロック（ブロック１）の消去可能でないデータを仮の退避領域（ブロック０）に移動して、移動元のブロックを消去する。そして、そのブロック（ブロック１）を退避領域として使用する。ついで、（ｉ）に示すように、予備ブロック（ブロック０）の退避動作を行う。すなわち、仮の退避領域として用いた予備領域（ブロック０）の論理セクタＢ，Ｃをブロック３に移動する。これで、予備領域と退避領域が作成されたので、（ｊ）において、論理セクタＡ，Ｍ，Ｎをブロック３に書き込む。次に、セクタＡ，Ｇ，Ｈ，Ｏ，Ｐを書き込もうとするが、書き込み領域がないので、図２９（ｋ－１）に示すように退避動作を行う。すなわち、消去可能なデータが最も多いブロック（ブロック４）の消去可能でないデータを退避領域（ブロック１）に移動して、移動元のブロックを消去する。そして、そのブロック（ブロック４）を退避領域として使用する。ついで、（ｋ－２）に示すように、セクタＡ，Ｇ，Ｈ，Ｏ，Ｐを書き込むが、セクタＡ，Ｇ，Ｈについては同じアドレスの論理セクタがあるので、消去フラグを立てたのちブロック１に書き込み、セクタＯ，Ｐについては、そのまま、書き込む。

【００３７】次に、（ｌ）において、セクタＡを書き込もうとするが、書き込み領域がないので、退避動作を行う。その結果、ブロック２が退避領域となり、ブロック２にあったセクタＤ，Ｅ，Ｆがブロック４に移動する。ここで、ブロック２が消失失敗で使用不可になったとすると、退避領域がなくなるので、救済措置を行う。すなわち、消去可能なデータが最も多いブロック（ブロック３）のデータを仮の退避領域（ブロック０）に移動して、移動元のブロックを消去する。そして、そのブロック（ブロック３）を退避領域として使用する。その結果、図３０（ｍ）に示すようになる。なお、この状態で、電力不足などにより、処理に一旦ストップをかけたすると、処理を再開したとき、装置側は予備領域に消去可能なセクタがなく、データがあるので、空き領域作成中と判断できる。続いて、空き領域作成のための救済措置を行い、ブロック６のセクタＩ，Ｊを予備領域（ブロック０）に移動し、ブロック６のセクタＫ，Ｌをブロック３に移動する。その結果図３０（ｎ）に示すようになる。

【００３８】ここで、退避領域のセクタを除いた空きセクタ数が、予備領域（ブロック０）にあるデータの数以上なので、書き込みポイントを退避領域の後ろから検索し、空いている領域に、予備領域（ブロック０）からデータを移動する。その際、図３１（ｏ）に示すように、予備領域（ブロック０）のセクタＢ，Ｃ，Ｍ，Ｎを移動した段階で、本体側が電力不足等により一旦処理にストップをかけたとする。この場合には、処理を再開した時に、装置側は予備領域に消去可能なセクタがあるので、空きブロック作成中の予備領域のデータ移動中と判断できる。続いて、図３１（ｐ）に示すように、予備領域（ブロック０）の論理セクタＩ，Ｊをブロック４に移動して処理を終了する。

【００３９】図３２は本実施例における空き領域作成処理を示すフローチャートであり、同図により、本実施例の処理を説明する。ステップＳ１において、予備領域に消去可能セクタが有るか否かを判別し、ある場合には、ステップＳ１１に行く。ない場合には、ステップＳ２に行き、書き込みポインタを予備領域の先頭とし、ステップＳ３において予備領域にデータがあるか否かを判別する。予備領域にデータがある場合には、ステップＳ１０に行き、ない場合にはステップＳ４に行き、退避領域を予備領域とする。ステップＳ５において、書き込み領域の各ブロックに消去可能セクタがない場合には、空き領域を作成できないので処理を放棄（Abort ）する。消去可能セクタがある場合には、ステップＳ６に行き、消去可能セクタの一番多いブロックを退避対象とし、ステップＳ７において、退避対象のブロックからデータを予備領域に移動する。

【００４０】ステップＳ８において退避対象となるブロックを消去し、ステップＳ９において、退避対象のブロックを退避領域にする。ステップＳ１０において、①予備領域のデータ数が②退避領域以外の空きセクタ数より多いか否か判別し、①＞②の場合には、ステップＳ５に戻り、上記処理を繰り返す。また、①≦②の場合には、ステップＳ１１に行き、書き込みポインタを退避領域の次から書き込み方向に対して検索し、ステップＳ１２に

おいて、空きセクタに予備領域からデータを移動する。ステップＳ１３において、予備領域を消去して終了する。本実施例においては、上記のように空き領域を作成しているので、記憶媒体中に使用不可のブロックが発生しても、救済措置により空き領域を作成することができ、記憶装置して使用不可能となることを避けることができる。また、処理途中で、本体側の電力不足などにより処理が中断しても、どの段階で処理が中断したかを容易に判断することができ、処理中断における処理段階の判断を単純化することが可能となる。

【００４１】Ｃ．消去処理の効率化

フラッシュ・メモリは前記したように上書きをすることができず、消去した後でなければ書き込みを行うことができない。本実施例は、上記のようなフラッシュ・メモリにおいて、書き込み有無を示すフラグと、全空間消去処理を実行中であることを示す実行中フラグを設けることにより、全記憶空間を消去する際の消去処理を効率良く行うとともに、消去中に何らかの原因で処理が中断された場合においても、処理再開をすることができる実施例を示している。次に図３３、図３４により本実施例の処理を説明する。なお、本実施例においは、ブロック数は６個とし、各ブロックに書き込みフラグを書き込む領域を設け、ブロック６に全空間消去処理を実行中であることを示す実行中フラグを書き込む領域を設けている。図３３（ａ）において、ブロック１，２，４の書き込み前に、ブロック１，２，４に書き込み有無フラグを書き込む。ついで、全空間をクリアする初期化処理に当たり、（ｂ－１）に示すように、ブロック６に実行中フラグを書き込む。（ｂ－２）において、書き込み有無フラグが書き込まれたブロック１を消去し、書き込む有無フラグを消す。

【００４２】図３３（ｂ－３）において、書き込み有無フラグが書き込まれたブロック２を消去し、書き込む有無フラグを消す。ここで、ブロック２の消去後、何らかの原因により、ストップ・シーケンスに移り、その後立ち上がった場合でも、ブロック６に実行中フラグが書き込まれているので、直ちに初期化動作中と判断し、初期化動作に移ることができる。（ｂ－４）において、ブロック１～３には書き込み有無フラグに書き込まれていないため、ブロック４を消去し、書き込む有無フラグを消す。（ｂ－５）において、ブロック１～５には書き込み有無フラグに書き込まれていないため、ブロック６を消去し、実行中フラグを消す。

【００４３】図３５は本実施例の処理を示すフローチャートであり、同図により本実施例を説明する。ステップＳ１において、実行中フラグの書き込みがあるか否かを判別し、有る場合には、ステップＳ３に行き、ない場合にはステップＳ２において、最終処理ブロックに実行中フラグを書き込む。ステップＳ３において、処理対象をクリアし、ステップＳ４において、処理対象に書き込み有無フラグがあるか否かを判別する。書き込み有無フラグがない場合には、ステップＳ６に行き、有る場合には、ステップＳ５において、処理対象を消去し、ステップＳ６において、処理対象を次に進める。ステップＳ７において、実行中フラグの書き込みかあるか否か判別し、ある場合には、ステップＳ４に戻り上記処理を繰り返す。また、実行中フラグがない場合には、終了する。本実施例においては、上記のように書き込み有無フラグを設けているので、不要な消去を避けることができ、消去時間を短縮することができるとともに、媒体の寿命を延ばすことができる。また、初期化実行中であることを示す実行中フラグを設けているので、不意の中断に対応することができる。

【００４４】Ｄ．書き込み速度の向上、および、不良セクタ発生時の予備領域の割り当て

前記したように、フラッシュ・メモリは上書きをすることができず、消去した後でなければセクタの書き込みを行うことができない。このため、前記したように、書き込み時、同一のセクタがある場合には消去可能フラグを立て、適宜の時点で消去可能フラグが立っているセクタを消去する必要があり、書込みの時間が掛かる。また、前記したように、フラッシュ・メモリは、消去回数に制限があり、所定回数以上消去を行うと消去が出来なくなる。このため、不良セクタが発生した場合には、予備の領域を割り当て救済措置を講ずる必要があり、以下に示す実施例は、上書きができず、セクタ単位で消去可能なメモリにおいて、上記した書き込み時のタイムラグをなくし、書き込み速度を向上するとともに、不良セクタが発生した場合に予備領域を割り当てる実施例を示している。図３６は本実施例のシステム構成を示す図であり、図５に示したシステムに、ＥＥＰＲＯＭ２６が付加されており、その他の構成は図５に示したものと同一である。ＥＥＰＲＯＭ２６にはアドレス変換を行うデコード・テーブルが格納されており、上記ＥＥＰＲＯＭ２６によりアドレスのデコードが行われる。そして、フラッシュ・メモリの一部が不良になったとき、ＥＥＰＲＯＭ２６を書き換えて、救済措置を行う。なお、図３６には、ＥＥＰＲＯＭ２６が例示されているが、デコード・テーブルを格納した記憶媒体としては、書き換えが可能な記憶媒体であればよく、その他フラッシュ・メモリ等を用いることができる。

【００４５】（１）実施例１

図３７、図３８は本実施例の構成を示す図であり、図３７において、２６１はＥＥＰＲＯＭ２６（もしくはフラッシュ・メモリ等の書き換え可能なＲＯＭ）から構成されるセクタ変換部である。２５はフラッシュ・メモリであり、フラッシュ・メモリは第１の領域Ａ、第２の領域Ｂ、および予備領域Ｃが設けられ、第１の領域Ａ、第２の領域Ｂには、それぞれ、１～４の４個のセクタが設けられている。図３７において、データをセクタ１に書き

込む場合には、第１の領域Ａと第２の領域Ｂを見て、既に第１の領域Ａにデータが有る場合には、第２の領域Ｂにデータを書き込む。また、上記データを書き込んでいる間に、第１の領域Ａのセクタ１－Ａを消去する。その際、書き込みエラーが発生した場合には、予備領域Ｃを代替セクタとして割り当てる。すなわち、第１の領域Ａのセクタ１－Ａに書き込みエラーが発生した場合には、予備領域Ｃをセクタ１－Ａの代替とするため、セクタ変換部２６１のＲＯＭを書き換え、図３８に示すように、セクタ１－Ａとして、予備領域Ｃを割り当てる。

【００４６】以上のように、書き込みエラーが発生した場合、予備がある限り予備領域のセクタを割り当て、予備がなくなった場合には、データを全て図３６に示したＳＲＡＭ２３等の外部の領域に退避し、セクタ変換部２６１のＲＯＭを再編成し、データの空間と予備の空間に分ける。また、その時、セクタは、システムの構成に応じて、昇順、または、降順で初期状態と同様に割り当てていくが、不良セクタは除外する。本実施例においては、上記のように、書き込みと同時に消去を行っているので、書き込み時のタイムラグをなくすことができ、書き込み速度を向上させることができる。また、予備領域を設け、不良セクタが発生した場合に、予備領域に代替しているので、不良セクタが発生した際の救済を行うことができる。

【００４７】（２）実施例２

図３９、図４０は本実施例の構成を示す図であり、図３９において、２６１はＥＥＰＲＯＭ、もしくは、フラッシュ・メモリ等の書き換え可能なＲＯＭから構成されるセクタ変換部、２５はフラッシュ・メモリであり、フラッシュ・メモリは第１の領域Ａ、予備領域Ｃが設けられ、第１の領域Ａには１～４の４個のセクタが設けられている。また、２３１は図３６に示したＳＲＡＭ２３等のデータの一次記憶媒体である。図３９において、データの書き込み時、一次媒体２３１にデータを転送している間に、書き込むセクタを調べ、書き込むべきセクタにデータがあった場合には、そのセクタを消去する。なお、転送が早く終了した場合には、消去終了までまち、また、転送が遅い場合には、転送終了後、データを書き込む。また、書き込みエラーが発生した場合には、セクタ変換部２６１のＲＯＭの書き換えをおこなって、図４０に示すように（セクタ１－Ａにエラーが発生した場合を示している）、予備領域Ｃにセクタ１－Ａの書き込みを行う。なお、通常のセクタがエラーにならない限り、予備領域Ｃへの書き込みは行わない。本実施例においては、上記のように、データを一次記憶媒体２３１に転送している間に、書き込むべきセクタにデータがあった場合、消去を行っているので、第１の実施例と同様、書き込み時のタイムラグをなくすことができ、書き込み速度を向上させることができる。また、予備領域を設け、不良セクタが発生した場合に、予備領域に代替しているので、不良セクタが発生した際の救済を行うことができる。

【００４８】（３）実施例３

図４１、図４２は本実施例の構成を示す図であり、図４１において、２６１はＥＥＰＲＯＭ、もしくは、フラッシュ・メモリ等の書き換え可能なＲＯＭから構成されるセクタ変換部、２５はフラッシュ・メモリであり、フラッシュ・メモリはブロック１，ブロック２，予備ブロック１，予備ブロック２が設けられ、各ブロックには、ブロック１，ブロック２には、セクタ１～３Ａ，１～３Ｂ，…，１～３Ｄ、４～６Ａ，４～６Ｂ，…，４～６Ｄが設けられている。そして、セクタ１～３Ａ，１～３Ｂ，…，１～３Ｄは物理アドレスとしてはそれぞれ１セクタであり、ここに論理アドレス１，２または３のデータを書き込むことができる。すなわち、論理アドレス１，２または３のデータを書き込む領域がＡ～Ｄの４セクタ分設けられており、データを書き込む場合には、Ａ～Ｄの内の空いている領域に書き込む。同様に、セクタ４～６Ａ，４～６Ｂ，…，４～６Ｄも上記と同様であり、論理アドレス４，５または６のデータを書き込む場合には、Ａ～Ｄの内の空いている領域に書き込む。以上のように、本実施例においては、ｎ＋１（この実施例においてはｎ＝３）セクタ単位でブロック化されており、書き込む論理アドレスに対して、１セクタの空きが設けられている。

【００４９】図４１において、データをセクタ１に書き込む際、セクタ１～３Ａ，１～３Ｂ，…，１～３Ｄを見て、これらの中にセクタ１が既に存在する場合には、そのセクタを消去する。また、セクタには必ず空いているセクタが存在するので、上記した消去と同時に、空いているセクタに書き込みを行う。書き込み時、書き込みエラーが発生した場合には、代替ブロックとして予備ブロックを割り当てる。例えば、セクタ１～３Ａが不良となった場合には、図４２に示すように、予備ブロック１を代替ブロックとして割り当てる。そして、セクタ変換部２６１のＲＯＭを書き換えて、エラーのあったブロック１を予備ブロック１に割り当てる代替処理を行う。これにより予備ブロック１がブロック１となる。予備がある限り上記のような処理を行い、予備がなくなった場合には、データを全て図３６に示したＳＲＡＭ２３等の外部の領域に退避し、セクタ変換部２６１のＲＯＭを再編成し、データの空間と予備の空間に分ける。また、その時、セクタは、システムの構成に応じて、昇順、または、降順で初期状態と同様に割り当てていくが、不良セクタは除外する。セクタ変換部２６１は、論理セクタから物理セクタへの変換時、上記したように、ｎ＋１セクタ単位でブロック化し、デコードするセクタ１に対して、書き込み可能なｎ＋１のセクタが存在するように構成する。そして、一部のセクタが不良となった場合には、上記セクタ変換部２６１のテーブルを書き換えるこ

とにより、書き込みブロックを移動して、常にｎ＋１のセクタのブロックを作成する。本実施例においては、上記のように、ｎ＋１セクタ単位でブロック化し、１セクタを空けておき、書き込みと同時に消去を行っているので、実施例１，２と同様、書き込み時のタイムラグをなくすことができ、書き込み速度を向上させることができる。また、予備領域を設け、不良セクタが発生した場合に、予備領域に代替しているので、不良セクタが発生した際の救済を行うことができる。

【００５０】（４）実施例４

図４３、図４４は本実施例の構成を示す図であり、図４３において、２６１はＥＥＰＲＯＭ、もしくは、フラッシュ・メモリ等の書き換え可能なＲＯＭから構成されるセクタ変換部、２５はフラッシュ・メモリであり、フラッシュ・メモリはブロック１，ブロック２，予備１Ａ～１Ｄが設けられ、ブロック１，ブロック２には、それぞれ、セクタ１～３Ａ，１～３Ｂ，…，１～３Ｄ，４～６Ａ，４～６Ｂ，…，４～６Ｄが設けられている。また、本実施例においては、実施例３と同様、ｎ＋１（この実施例においてはｎ＝３）セクタ単位でブロック化し、１セクタを空けておく。図４３において、データをセクタ１に書き込む際、セクタ１～３Ａ，１～３Ｂ，…，１～３Ｄを見て、これらの中にセクタ１が既に存在する場合には、そのセクタを消去する。また、セクタには必ず空いているセクタが存在するので、上記した消去と同時に、空いているセクタに書き込みを行う。書き込み時、書き込みエラーが発生した場合には、代替セクタとして予備１Ａ～１Ｄを割り当てる。例えば、セクタ１～３Ａが不良となった場合には、図４４に示すように、予備１Ａ～１Ｄがブロック１となり、セクタ１～３Ａに対して予備１Ａを割り当て、セクタ変換部２６１のＲＯＭを書き換えて代替処理を行う。これにより予備セクタ１Ａがセクタ１～３Ａとなり、予備１Ｂ～１Ｄはブロック１の予備となる。

【００５１】予備がある限り上記のような処理を行い、予備がなくなった場合には、データを全て図３６に示したＳＲＡＭ２３等の外部の領域に退避し、セクタ変換部２６１のＲＯＭを再編成し、データの空間と予備の空間に分ける。また、その時、セクタは、システムの構成に応じて、昇順、または、降順で初期状態と同様に割り当てていくが、不良セクタは除外する。セクタ変換部２６１は、論理セクタから物理セクタへの変換時、上記したように、ｎ＋１セクタ単位でブロック化し、デコード時には、そのデコード・データよりアドレス変換するテーブルを設けセクタ１に対して書き込み可能なｎ＋１のセクタが存在するように構成する。そして、一部のセクタが不良となった場合には、上記セクタ変換部２６１のテーブルを書き換えることにより、常にｎ＋１のセクタのブロックを作成する。本実施例においては、上記のように、ｎ＋１セクタ単位でブロック化し、１セクタを空けておき、書き込みと同時に消去を行っているので、実施例１，２，３と同様、書き込み時のタイムラグをなくすことができ、書き込み速度を向上させることができる。また、予備領域を設け、不良セクタが発生した場合に、予備領域に代替しているので、不良セクタが発生した際の救済を行うことができる。

【００５２】（５）実施例５

図４５、図４６は本実施例の構成を示す図であり、図４５，４６において、図３７、図３８に示したものと同一のものには同一の符号が付されており、２６１はＥＥＰＲＯＭ、もしくは、フラッシュ・メモリ等の書き換え可能なＲＯＭから構成されるセクタ変換部、２５はフラッシュ・メモリであり、フラッシュ・メモリはブロック１，ブロック２，予備ブロック１，予備ブロック２が設けられ、ブロック１，ブロック２、セクタ１～３Ａ，１～３Ｂ，…，１～３Ｄ，４～６Ａ，４～６Ｂ，…，４～６Ｄが設けられている。また、本実施例においては、実施例３と同様、ｎ＋１（この実施例においてはｎ＝３）セクタ単位でブロック化し、１セクタを空けておく。図４５において、データをセクタ１に書き込む際、セクタ１～３Ａ，１～３Ｂ，…，１～３Ｄを見て、これらの中にセクタ１が既に存在する場合には、そのセクタを消去する。また、必ず空いているセクタが存在するので、上記した消去と同時に、空いているセクタに書き込みを行う。書き込み時、書き込みエラーが発生した場合には、代替セクタとして予備セクタを割り当てる。例えば、セクタ１～３Ａが不良となった場合には、予備ブロック１を割り当て、セクタ変換部２６１のＲＯＭを書き換えて、代替処理を行う。これにより、図４６に示すように、予備ブロックのセクタ１，ｎ，ｍＡがセクタ１に置き変わり、予備ブロックにセクタ１が存在することとなり、元のブロックにはセクタ２，３がそのまま残る。

【００５３】すなわち、この実施例においては、ブロックの構成は続きセクタでなくてもよく、元のブロックの構成数を変更する。予備がある限り上記のような処理を行い、予備がなくなった場合には、データを全て図３６に示したＳＲＡＭ２３等の外部の領域に退避し、セクタ変換部１１のＲＯＭを再編成し、データの空間と予備の空間に分ける。また、その時、セクタは、システムの構成に応じて、昇順、または、降順で初期状態と同様に割り当てていくが、不良セクタは除外する。本実施例においては、上記のように、ｎ＋１セクタ単位でブロック化し、１セクタを空けておき、書き込みと同時に消去を行っているので、実施例１，２，３，４と同様、書き込み時のタイムラグをなくすことができ、書き込み速度を向上させることができる。また、予備領域を設け、不良セクタが発生した場合に、予備領域に代替しているので、不良セクタが発生した際の救済を行うことができる。

【００５４】図４７～図４９は上記実施例における処理を示すフローチャートであり、図３６に示したシステム

における本実施例の処理を説明する。ステップＳ１において、本体からセクタデータの　き込み要求があると、ステップＳ２において、ＣＰＵ２２はコントローラＬＳＩ２１からの通知によりデータの書き込み要求があることを認識する。なお、コントローラＬＳＩ２１がＣＰＵ２２に割り込みをかけて、データの書き込み要求を通知してもよい。ステップＳ３において、ＣＰＵ２２はＥＥＰＲＯＭ２６のデータを読み込み、その書き込み場所を認識し、セクタデータが書かれているか否かをチェックするため、フラッシュ・メモリ２５－１，…，２５－５をアクセスし、ＥＣＣデータ、あるいは、フラッシュ・メモリに格納された管理情報等を参照する。そして、ステップＳ５において、フラッシュ・メモリにデータが書かれているか否かを判断し、データが書かれていない場合には、図４８のステップＳ８に行く。また、データが書かれている場合には、ステップＳ６において、第２のセクタにデータを書き込むための情報を読み込むため、ＣＰＵ２２はＥＥＰＲＯＭ２６をアクセスし、データを得る。ついで、ステップＳ７において、フラッシュ・メモリのデータを消去し（効率を上げるためエラーチェックは後で行う）、図４８のステップＳ８に行く。

【００５５】ステップＳ８において、本体よりデータをＳＲＡＭ２３にデータを転送し、終了を待つ。ステップＳ９において、フラッシュ・メモリにデータを書き込むため、ＣＰＵ２２はコントローラＬＳＩ２１に対してデータの書き込みを通知し、コントローラＬＳＩ２１はＳＲＡＭ２３よりフラッシュ・メモリにデータを書き込む。ついで、ステップＳ１０において、ＣＰＵ２２はコントローラＬＳＩ２１よりデータ書き込み終了の通知を受け、ステップＳ１１において、データの書き込みエラーがあったか否かを判断する。データの書き込みエラーがなかった場合には終了し、書き込みエラーがあった場合には、ステップＳ１２において、代替用セクタが有るか否かを判断し、代替用セクタがある場合には、ステップＳ１３に行き、セクタアドレスを変更するため、ＥＥＰＲＯＭ２６をアクセスしてデータを変更し、ステップＳ９に戻る。また、代替用セクタがない場合には、図４９のステップＳ１４に行き、消去中のフラッシュ・メモリがあるか否かを判断し、消去中のフラッシュ・メモリがある場合には、ステップＳ１５に行き、消去終了まで待って、消去したセクタにデータを書き込み終了する。ステップＳ１４において、消去中のフラッシュ・メモリがないと判断された場合には、ステップＳ１６において、本体側にエラーを通知する。本体側では、ステップＳ１７において、データを吸い上げＥＥＰＲＯＭ２６を再編成するコマンドを送る。記憶装置側では、本体側からのコマンドに応じて、ＥＥＰＲＯＭ２６を再編成し、本体から送られる１論理セクタに対して、２セクタ以上の選択ができるようにする。これにより、少し容量は減るが記憶装置として使用可能となる。なお、上記処理においては、ＳＲＡＭ２３にデータを転送し、フラッシュ・メモリにデータを　き込む場合について説明したが、ＳＲＡＭ２３を設けず、直接フラッシュ・メモリにデータを書き込むこともできる（この場合には、上記フローチャートのステップＳ８，Ｓ９は一処理となる）。

【００５６】Ｅ．アドレス変換用デコード部の信頼性の向上

以下、本発明の請求項１の発明の実施例について説明する。論理アドレスを物理アドレスに変換するための変換テーブルとしてＲＯＭが使用されることがある。また、ＲＯＭの代わりに、フラッシュＥＥＰＲＯＭ、ＥＥＰＲＯＭ等を使用することも可能であるが、通常はデコード部を可変にする必要がないため、使用されることが少ない。一方、フラッシュ・メモリは、その媒体の性質上、消去回数に制限があり、不良になると、不良になったアドレスが使用出来なくなる。そこで、前記したＤ．の実施例（１）～（５）に示すように、論理アドレスを物理アドレスに変換するための変換テーブルとして、フラッシュＥＥＰＲＯＭ、ＥＥＰＲＯＭなどの書き換え可能なデコード部を用い、不良のセクタが発生した場合、デコード部を書き換え、不良アドレスがないように見せかける方法が用いられる。しかしながら、変換テーブルとして、フラッシュＥＥＰＲＯＭ、ＥＥＰＲＯＭなどを用いた場合、これらのフラッシュＥＥＰＲＯＭ、ＥＥＰＲＯＭなどにも、消去回数に制限があり、所定回数以上消去を行うと不良が発生する。本実施例は、上記のように、デコード部としてフラッシュＥＥＰＲＯＭ、ＥＥＰＲＯＭなどの書き換え可能な記憶媒体が使用されている場合において、デコード部を２段化とすることにより、デコード部の寿命を延ばし、アドレス変換用デコード部の信頼性を向上させる実施例を示している。

【００５７】図５０は本実施例のデコード部の構成を示す図であり、同図（ａ）は通常の場合、（ｂ）はセクタが不良になった場合、（ｃ）はデコード部が不良になった場合を示しており、３０１は１次デコード部、３０２は２次デコード部、３０３はフラッシュ・メモリのセクタを示している。同図において、フラッシュ・メモリのセクタが正常な場合には、（ａ）に示すように、１次デコード部３０１がセクタ０のアドレスのデコード値としてアドレス「０００ｈ」を出力し、２次デコード部３０２が「０００ｈ」のデコード値として物理アドレス「５５５５ｈ」を出力し、フラッシュ・メモリのセクタ０の物理アドレスを指定している。ここで、フラッシュ・メモリの消去回数が制限値を越えてセクタが不良となり、セクタ０の物理アドレスをアドレス８８８８ｈの予備のセクタに切り替える場合には、同図（ｂ）に示すように、２次デコード部３０２を書き換えて、「０００ｈ」のデコード値を「５５５５ｈ」から「８８８８ｈ」に切り替える。これにより、セクタ０として物理アドレス「８８８８ｈ」が割り当てられる。上記のようにし

て、２次デコード部（もしくは１次デコード部）を何度か書き換えることにより書き換え制限値以上になり、２次デコード部に不良になると、同図（ｃ）に示すように、一次デコード部３０１を書き換えてそのデコード値をずらし、２次デコード部３０２がデコード値として「８８８８ｈ」を出力する「２２２２ｈ」をデコード値として出力するように書き換える。これにより、２次デコード部３０２が不良になっても、物理アドレス「８８８８ｈ」をデコード値として出力することができる。

【００５８】以上のように、デコード部を２段化し、一方のデコード部が不良になった場合に、他方のデコード部を書き換えることにより、デコード部の寿命を２乗化することができる。例えば、セクタが不良になるまでの回数をＬ回（書き換え制限値）、２次デコード部が不良になるまでの回数をＭ回（書き換え制限値）、１次デコード部が不良になるまでの回数をＮ回（書き換え制限値）とすると、アドレスがエラーになるまでの回数はＮ×Ｍ×Ｌ回となる。基本的に書き換え制限値はフラッシュＥＥＰＲＯＭで１０万～１００万回、ＥＥＰＲＯＭで１万回程度といわれているので、事実上、デコード部を２段化するだけで、信頼性を十分確保することができる。

【００５９】Ｆ．書き込み時間の推定処理

フラッシュ・メモリにおいては、データの書き込みに際、退避処理等を行う必要があり、通常の半導体メモリの場合より書き込みに時間がかかる。本実施例は、フラッシュ・メモリへの書き込み時間の推定を行い、消費電力の推定、異常の検出等を行う実施例を示している。図５１に本実施例のシステム全体の概略構成を示す。１は記憶装置、１ａは上書きすることができず、また、セクタ、ブロック等の所定の単位でしかデータを消去することができない、例えばフラッシュ・メモリ等から構成される記憶領域、１ｂはコントローラ、２は本体処理装置である。次に本実施例について説明する。今、記憶装置が図５２に示すような状態になっている場合に、３セクタのデータを書き込む場合を考える。まず、図５１の本体よりプロセッサ２２に書き込むセクタ数を送る。書き込むセクタ数がプロセッサ２２に送られると、プロセッサは書き込むセクタ数により書き込み時間を求める。ここで、図５２のように書き込むブロックが無い場合には、退避動作が行われるので、書き込み時間ｔは次のようにして求められる。

ｔ＝〔１セクタの書き込みに掛かる時間〕×３＋〔データの退避時間〕（秒）

プロセッサ２２は上記のようにして書き込み時間を計算し、本体へ返す。本体は、上記書き込み時間を元に、次の式により書き込み動作時の消費電力Ｗ１を求める。

Ｗ１＝ｔ÷３６００×〔書き込み動作の消費電力〕（Ｗ）

また、本体はその電源部から残りの電力Ｗ２を読み取り、得られた書き込み電力と比較し、Ｗ２＞Ｗ１ならデータの書き込みを行う。一方、本体は上記のようにして得た書き込み時間ｔと実際の書き込み時間を比較し、上記書き込み時間ｔを越えても書き込みが終了しない場合には、記憶装置の異常と判断し、例えば、ユーザに通知したり、書き込み処理をストップ等の措置を行う。

【００６０】図５３は本実施例における本体の処理を示すフローチャートであり、同図により本実施例を説明する。ステップＳ１において、これから書き込むデータのサイズを記憶装置に送り、書き込みにかかる時間を得る。ステップＳ２において、書き込みにかかる時間と、時間当たりの消費電力をかけて書き込みにかかる電力を得る。ステップＳ３において、残り電力と書き込みで使用する電力を比較し、残り電力の方が少ない場合には、異常処理を行う。また、残り電力の方が多い場合には、ステップＳ４に行きデータを書き込み、ステップＳ５において、実際の書き込み時間が予想した時間より多いか否か判別し、多い場合には、異常処理を行う。また、書き込み時間が予想した時間以内の場合には、ステップＳ６に行き、書き込み終了か否かを判別し、書き込み終了でない場合にはステップＳ５に戻り、書き込み終了の場合には、終了する。

【００６１】以上のように、本実施例においては、書き込み時間を求めることにより、書き込みに要する電力を推定しているので、残り電力量で書き込みができるかどうかの判断を行うことができる。このため、電池駆動等のシステムにおいて、書き込み中に電力がなくなり書き込みが途中で中断することを避けることができ、システムの信頼性を向上させることができる。また、実際の書き込み時間と推定した書き込み時間を比較することにより、記憶装置の異常を検出することができ、また、書き込み時間を表示するなどしてユーザに知らせることにより、ユーザが記憶装置の異常を判断することが可能となる。

【００６２】Ｇ．フラグ判定処理における信頼性の向上

フラグの判定は、通常、１ビットを１機能に割り当て判定している場合が多い。しかしながら、フラッシュ・メモリは過剰消去により消去不能になるという、致命的不良原因を構造的にもっており、これが発生したセルはハイレベルのままで、ローレベルに戻らない状態に陥る。このため、フラッシュ・メモリ上のフラグを１ビット１機能に対応させた場合には、そのビットが不良になったとき、フラグの有効性がなくなってしまう。本実施例は、前記図５１に示したようにフラッシュ・メモリ上の消去フラグ、不良フラグ、バリテイー等に対して、フラグを複数用意し論理積によりフラグ判定することにより、上記したフラッシュ・メモリの構造的欠点に対処し、フラグ判定の信頼性を向上させた実施例を示している。

【００６３】（１）実施例１

図５４は実施例１を示す図であり、同図（ａ）はフラグレジスタ、（ｂ）は判定のための論理積回路、（ｃ）は真理値表を示している。本実施例においては、同図（ａ）に示すように、一つの機能に対してフラグｂ０，ｂ４を用意してフラグレジスタに格納する。そして、フラグ判定を行う場合には、（ｂ）に示すように、これらのフラグの論理積を求めることにより、最終的な判定結果を得る。上記のように判定しているので、同図（ｃ）の行番号２のようにビットｂ０がハイレベルで固定になっても、その論理積結果としては、正しい結果を得ることができる。同様に、行番号３のように、ビットｂ４がハイレベルで固定になっても、その論理積結果としては、正しい結果を得ることができる。

【００６４】（２）実施例２

図５５は実施例２を示す図であり、同図（ａ）は第１のフラグレジスタ、（ｂ）は第２のフラグレジスタ、（ｃ）はフラグ判定を行う論理積回路を示している。本実施例は同一機能に対するフラグを別々のフラグレジスタに格納するようにしたものであり、同図（ａ）に示す第１のフラグレジスタのビットｂ０にフラグを格納し、同図（ｂ）に示す第２のフラグレジスタのビットｂ０とビットｂ２に第１のフラグレジスタに格納したフラグと同一の機能に対応するフラグを格納している。そして、フラグを判定する場合には、同図（ｃ）に示す論理積回路により上記フラグの論理積を求める。本実施例においても、実施例１と同様、フラグの一部がハイレベルで固定しても、正しい判定結果を得ることができる。以上のように、本実施例においては、フラグの判定を複数ビットで行うようにしているので、全てのビットが不良にならない限り、正しい判定値を常に出力することができ、フラグ判定の信頼性を向上することができる。

【００６５】Ｈ．管理テーブルの節約

本実施例は記憶装置が複数チップで構成され、各チップ内に複数のブロックが設けられたフラッシュ・メモリ・システムにおいて、管理テーブルを簡単化するとともに、各ブロックを平均的に使用することができる実施例を示している。以上示した実施例においては、データを書き込むチップ、ブロックに制約がなく、データは全ブロックのセクタに書き込むことができたため、全領域を管理するテーブルを設ける必要があったが、本実施例においては、各チップ内に書き込むデータを固定とし、チップ内において書き込むブロックを移動して書き込んでいくことにより、全領域を管理することなくデータの書き込みができるようにし、フラッシュ・メモリの管理を簡単化するとともに、各ブロックの使用を平均化している。また、本実施例においては、チップ内に書き込まれるデータをチップの全容量より一定量少なくし、この一定量のブロックをワークブロックと不良ブロック発生時の救済ブロックに当てている。そして、ブロック内のデータの並びは固定とし、データを書き換える場合、すでに旧ブロックに有効なデータが存在している場合には、そのデータを前記した図５のＳＲＡＭ２３に退避して、新データと一緒にして空いている領域に書き込み、書き込みが成功したら旧データのブロックを消去するようにしている。

【００６６】次に本実施例を説明する。図５６は本実施例におけるチップ、ブロック、セクタ内構成を示す図であり、同図に示すように、チップは2Mbyteであり、チップ内は１ブロック4KbyteのＮｏ．０００～Ｎｏ．５１１の５１２個のブロックに分割され、各ブロック内には、１セクタ528byte のＮｏ．００～Ｎｏ．０７の８個のセクタが設けられている。なお、同図のブロックＮｏ．およびセクタＮｏ．は物理アドレスを示している。さらに、セクタ内には、同図に示すように、256byte の実データ領域、Ｌｏｎｇ命令用のＥＣＣ領域、不良データフラグ、不良ブロックフラグ、ブロックアドレス、256byte の実データ用領域ＥＣＣ領域が設けられている。ブロック・アドレスは、ＳＲＡＭ２３に格納されているブロック・ポインタのアドレス値を示し、ブロック・アドレスには、実データなしと判断したセクタも書き込むこととしている。このようにすることにより、活線抜き差し時、セクタアドレス数が多い物が正常データと判断することができる。なお、セクタアドレス数が等しい場合はＥＣＣ・チェックサムの正誤で判断する。不良ブロックフラグは、ブロックのコンディションを示しており、ＦＦｈで正常ブロック、≠ＦＦｈで不良ブロックとしている。不良データフラグは、データのコンディションを示しており、ＦＦｈで正常データ、≠ＦＦｈで不良データとしている。Ｌｏｎｇ命令用のＥＣＣ領域は最大4byteである。ＥＣＣ領域は、実データのコンディションを示しており、この領域の値によりデータの訂正および誤り検出を行う。

【００６７】図５７～図６１は本実施例における書き込み処理を示す図であり、同図により本実施例を説明する。本実施例においては、チップがＮｏ．００～Ｎｏ．０４の５個で、各チップにワークブロックを４個設けた場合の書き込みおよび管理方式を示している。図５７は全ブロックを消去した状態を示しており、同図に示すように、各チップにはワークブロックがｗｏｒｋ０１～０４の４個設けられており、データは上記ワークブロックｗｏｒｋ０１～０４に書き込まれる。書き込みはセクタ単位で可能であるが、セクタ００５の書き込み後にセクタ００４を書き込むなど、上位のセクタの書き込み後には下位のセクタの書き込みはできない。また、チップを越えてデータが移動することはなく、ブロック内のセクタの順番は不動である。さらに、例えば、論理ブロックアドレス００にセクタ０００～００５のみを書き込む場合のように、書き込みデータが８セクタ全てを満たさない場合には、セクタ０００～００５のデータを書き込み後、セクタ００６，００７にブロックアドレスのみを書

き込む。なお、この場合、１ブロックが８セクタであり、また、チップが５個であるので、論理アドレスｎのデータは、（論理アドレスｎ÷８）の商をさらに５で割ったときに得られた余りのチップに書き込まれる。例えば、ｎ＝１２１の場合には、１２１÷８は１５で余り１であり、１５÷５は５で余りが０なので、Ｎｏ．００のチップに書き込まれる。

【００６８】本実施例におけるデータの書き込み処理は次のように行われる。まず、図５７に示す全ブロックを消去した状態からセクタＮｏ．０００～０３９（論理アドレス）の４０セクタを連続書き込みすると、図５８に示すように各チップのｗｏｒｋ０１にデータが書き込まれる。すなわち、最初に書き込まれるセクタＮｏ．０００～００７は論理ブロックアドレスが００なので、本体からＳＲＡＭ２３にセクタＮｏ．０００～００７のデータを転送して、チップＮｏ．００の書き込みポインタが指すｗｏｒｋ０１からデータを書き込む。なお、論理ブロックアドレスはチップ単位で管理される。そして、この場合には論理ブロックアドレス００が過去の存在しないので、チップＮｏ．００のｗｏｒｋ０１はｗｏｒｋ０４の次の空き領域に移動する。８セクタ分の書き込みが終了したら、書き込みポインタをチップＮｏ．０１のｗｏｒｋ０１に移動し、上記と同様にセクタＮｏ．００８～０１５のデータをｗｏｒｋ０１に書き込む。

【００６９】以上のようにして、書き込みがチップＮｏ．０４まで完了したら、書き込みポインタをチップＮｏ．００のｗｏｒｋ０２に移動する。次に、図５８の状態から、図５９に示すようにセクタＮｏ．１２０～１９１（論理アドレス）の７２セクタの連続書き込みをする。上記と同様、本体からＳＲＡＭ２３にセクタＮｏ．１２０～１２７のデータを転送して、セクタＮｏ．１２０～１２７は論理ブロックアドレスが０３なので、チップＮｏ．００の書き込みポインタが指すｗｏｒｋ０２からデータを書き込む。そして、上記と同様、チップＮｏ．００のｗｏｒｋ０２、ｗｏｒｋ０３はｗｏｒｋ０１の次の空き領域に移動する。以下同様に、セクタＮｏ．１２８～１９１のデータをチップＮｏ．００～Ｎｏ．０４に書き込む。ここで、図５９の状態から、セクタＮｏ．００２～００３の２セクタを連続書き込みすると、図６０に示すようになる。まず、ＳＲＡＭ２３にセクタＮｏ．００２～００３のデータを転送する。この場合、書き込みデータがセクタＮｏ．００２～００３なので、論理ブロックアドレス００が指定され、これは過去に存在するので、旧データの退避が必要と判断される。また、書き込みデータがセクタＮｏ．００２～００３で、論理ブロックアドレス００の最初から書き込まれないので、旧論理ブロックアドレス００のデータをＳＲＡＭ２３に退避する。なお、書き込みデータが論理ブロックアドレスの先頭より２セクタ分のときには、書き込みデータを先に処理した後に旧論理ブロックアドレスのデータの退避動作を行う。

【００７０】次に、ＳＲＡＭ２３に退避した旧データ０００，００１，００４，００５，００６，００７と新データ００２，００３を書き込みポインタが指すチップＮｏ．００のｗｏｒｋ０４に書き込む。その際、書き込みは０００，００１，００２，００３，００４，００５，００６，００７の順番に書き込む。ついで、論理ブロックアドレス００がすでに存在しているので、旧論理ブロックアドレスの消去を行う。また、ｗｏｒｋ０４を旧論理ブロックアドレス００に移動する。上記のように２セクタ連続書き込みを行ったのち、電源がＯＦＦになり、再びＯＮにすると、図６１に示すように、ワークブロックは使われているブロックの次より指定される。また、上記のようにワークブロックを指定していき、チップの最後まで検索してもワークブロックを全部指定できなくなった場合には、チップの最初に戻り、ワークブロックを指定する。上記の処理において、ブロック消去エラーが発生した場合には、不良フラグに００ｈを設定し、そのブロックは書き込みおよび読み出し不可とする。すなわち、ワークブロックを一つ潰すことにより対処し、ワークブロックがなくなったら、そのシステムは書き込み不可とする。また、データ退避途中にエラーが発生した場合には、不良データフラグを立てたのちにデータを書き込む。なお、読み出し動作ではそのブロックにエラーが検出されても、そのブロックを不良ブロックとしては扱わない。データ書き込みエラーが発生した場合には、そのブロックを不良ブロックとする。不良フラグの立て方は、一度書かれているデータを退避し、不良ブロックフラグのみを立てて、再度書き込む。不良ブロックフラグも通常書き込みと同様、該当ブロック全てのセクタに対して立てる。データ消去途中でエラーが検出された場合には、そのブロックは不良ブロックとする。不良ブロックフラグの立て方は、不良ブロックフラグのみを立てるのではなく、実データを含め全てＡＬＬ〔００〕を書き込む。不良処理は該当ブロック全てのセクタに対して行う。

【００７１】以上のように、本実施例においては、各チップ内に書き込むデータを固定とし、チップ内において書き込むブロックを移動して書き込んでいくので、全領域を管理する必要がない。このため、データを各チップ内でのみ管理すればよく、管理テーブルを節約することができ、また、書き込み動作の高速化を図ることができる。また、ワークブロックにデータを書き込み、ワークブロックをチップ内で移動させるように構成したので、各ブロックを平均的に使用することができる。

【００７２】

【発明の効果】以上説明したように、本発明においては、デコーダを２段設け、記憶装置の記憶領域の一部が破壊したとき、もしくは、デコータの一部が破壊したとき、２段のデコーダのいずれか一方、もくしは、両方を

書き換えることにより、破壊した部分へのデコードが行われないようにしたので、デコーダとして、ＥＥＰＲＯＭ、フラッシュメモリ等のその一部が破壊する可能性のある記憶媒体を用いた場合においても、アドレスがエラーとなる確率を著しく減少させることができ、信頼性を確保することができる。

【図面の簡単な説明】

【図１】本発明の全体の概略構成を示す原理図である。

【図２】本発明の概要を示す原理図（１）である。

【図３】本発明の概要を示す原理図（２）である。

【図４】本発明の概要を示す原理図（３）である。

【図５】本発明の前提となる記憶装置の構成を示すブロック図である。

【図６】記憶装置におけるＳＲＡＭの内容を示す図である。

【図７】記憶装置におけるフラッシュ・メモリの内容を示す図である。

【図８】記憶装置におけるフラッシュ・メモリの内容を示す図（続き）である。

【図９】記憶装置における書き込み時の処理を示すフローチャートである。

【図１０】記憶装置における書き込み時の処理を示すフローチャート（続き）である。

【図１１】記憶装置における書き込み時の処理を示すフローチャート（続き）である。

【図１２】記憶装置における書き込み時の処理を示すフローチャート（続き）である。

【図１３】記憶装置における書き込み時の処理を示すフローチャート（続き）である。

【図１４】消去回数のバラツキを抑える実施例を示す図である。

【図１５】消去回数のバラツキを抑える実施例を示す図（続き）である。

【図１６】消去回数のバラツキを抑える実施例を示す図（続き）である。

【図１７】消去回数のバラツキを抑える実施例を示す図（続き）である。

【図１８】消去回数のバラツキを抑える実施例を示す図（続き）である。

【図１９】消去回数のバラツキを抑える実施例の処理を示すフローチャートである。

【図２０】消去可能データをなくす処理の実施例を示す図である。

【図２１】消去可能データをなくす処理の実施例を示す図（続き）である。

【図２２】消去可能データをなくす処理の実施例を示す図（続き）である。

【図２３】消去可能データをなくす処理の実施例を示す図（続き）である。

【図２４】消去可能データをなくす処理の実施例を示すフローチャートである。

【図２５】空き領域の作成処理の実施例を示す図である。

【図２６】空き領域の作成処理の実施例を示す図（続き）である。

【図２７】空き領域の作成処理の実施例を示す図（続き）である。

【図２８】空き領域の作成処理の実施例を示す図（続き）である。

【図２９】空き領域の作成処理の実施例を示す図（続き）である。

【図３０】空き領域の作成処理の実施例を示す図（続き）である。

【図３１】空き領域の作成処理の実施例を示す図（続き）である。

【図３２】空き領域の作成処理の実施例のフローチャートである。

【図３３】消去処理の実施例を示す図である。

【図３４】消去処理の実施例を示す図（続き）である。

【図３５】消去処理の実施例のフローチャートである。

【図３６】書き込み速度の向上を図る実施例のシステム構成を示す図である。

【図３７】書き込み速度の向上を図る第１の実施例を示す図である。

【図３８】書き込み速度の向上を図る第１の実施例を示す図（続き）である。

【図３９】書き込み速度の向上を図る第２の実施例を示す図である。

【図４０】書き込み速度の向上を図る第２の実施例を示す図（続き）である。

【図４１】書き込み速度の向上を図る第３の実施例を示す図である。

【図４２】書き込み速度の向上を図る第３の実施例を示す図（続き）である。

【図４３】書き込み速度の向上を図る第４の実施例を示す図である。

【図４４】書き込み速度の向上を図る第４の実施例を示す図（続き）である。

【図４５】書き込み速度の向上を図る第５の実施例を示す図である。

【図４６】書き込み速度の向上を図る第５の実施例を示す図（続き）である。

【図４７】書き込み速度の向上を図る実施例のフローチャートである。

【図４８】書き込み速度の向上を図る実施例のフローチャート（続き）である。

【図４９】書き込み速度の向上を図る実施例のフローチャート（続き）である。

【図５０】アドレス変換用デコード部を２段とした実施例を示す図である。

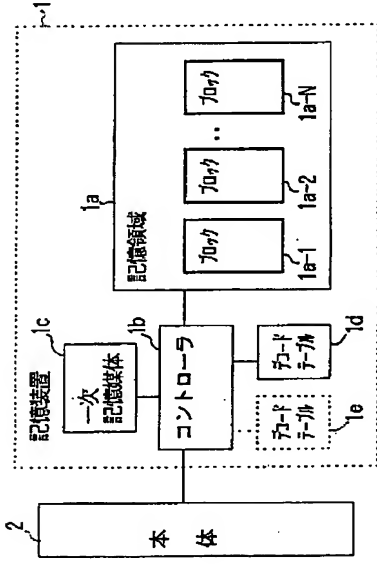
【図５１】書き込み時間の推定処理を行うシステム全体の概略構成を示す図である。
【図５２】書き込み時間推定処理の実施例における記憶領域の状態を示す図である。
【図５３】書き込み時間推定処理の実施例のフローチャートである。
【図５４】フラグ判定処理の第１の実施例を示す図である。
【図５５】フラグ判定処理の第２の実施例を示す図である。
【図５６】管理テーブルを節約する実施例における記憶領域の構成を示す図である。
【図５７】管理テーブルを節約する実施例を示す図である。
【図５８】管理テーブルを節約する実施例を示す図（続き）である。
【図５９】管理テーブルを節約する実施例を示す図（続き）である。
【図６０】管理テーブルを節約する実施例を示す図（続き）である。
【図６１】管理テーブルを節約する実施例を示す図（続き）である。

【符号の説明】

１，２０	記憶装置
１ａ	記憶領域
１ｂ	一次記憶媒体
１ｃ	制御手段
１ｄ，１ｅ	デコード・テーブル
２	本体処理装置
２１	コントローラＬＳＩ
２２	プロセッサ
２３	ＳＲＡＭ
２４	クロック発振器
２５－１～２５－５	フラッシュ・メモリ
２６	ＥＥＰＲＯＭ
２６１	セクタ変換部
３０１	１次デコード部
３０２	２次デコード部

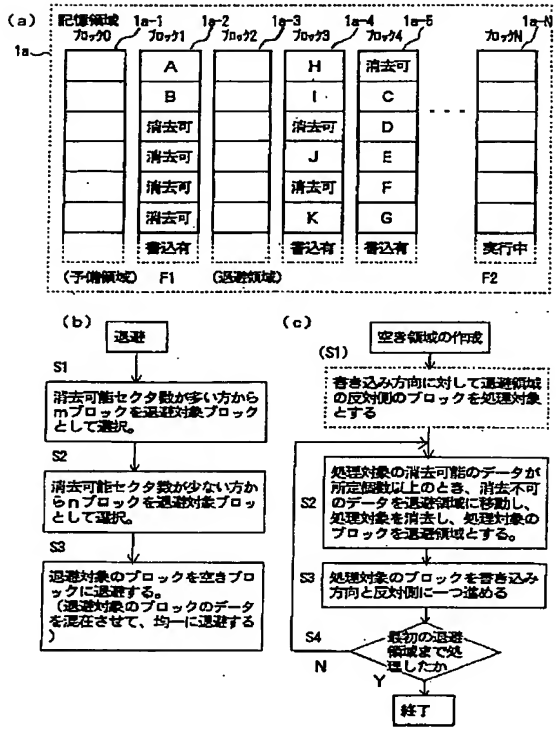
【図１】

本発明の全体の概略構成を示す原理図



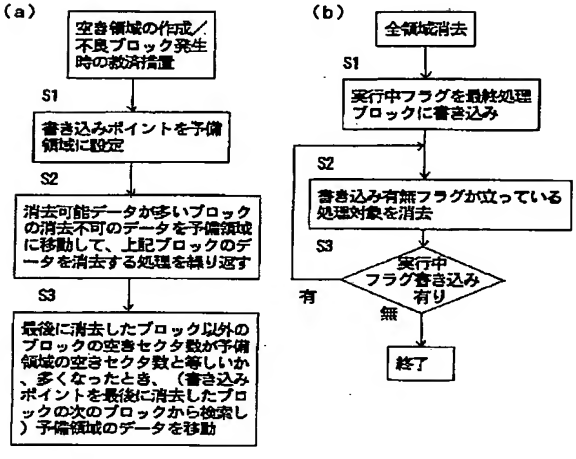
【図2】

本発明の原理を示す原理図（1）



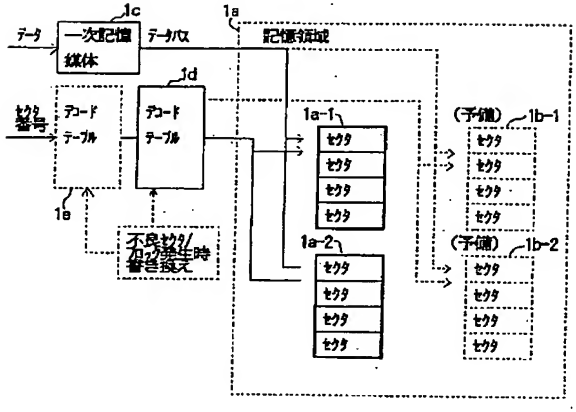
【図3】

本発明の原理を示す原理図（2）



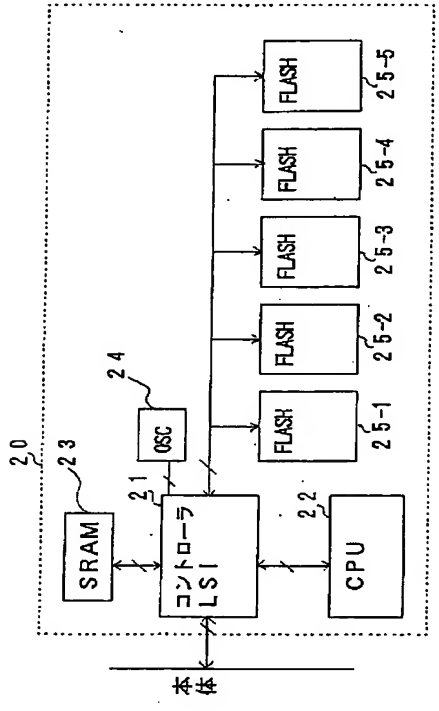
【図4】

本発明の原理を示す原理図（3）



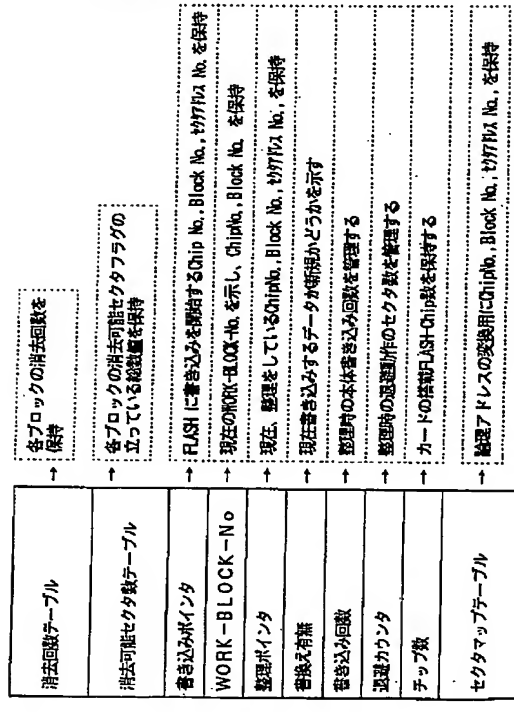
【図5】

本発明の前提となる記憶装置の構成を示すブロック図



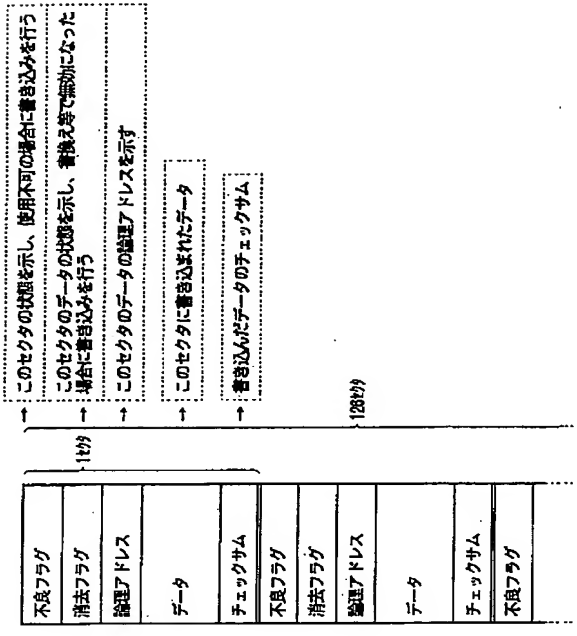
【図6】

記憶装置におけるＳＲＡＭの内容を示す図



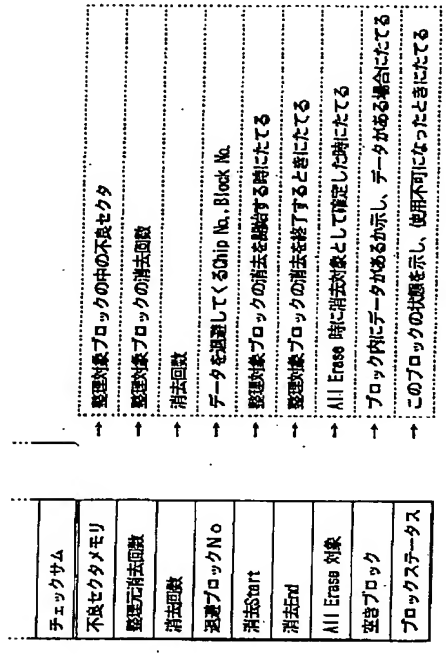
【図7】

記憶装置におけるフラッシュ・メモリの内容を示す図



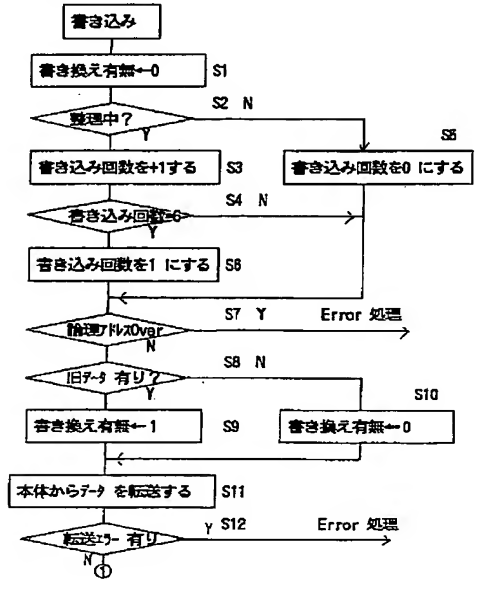
【図8】

記憶装置におけるフラッシュ・メモリの内容を示す図（続き）



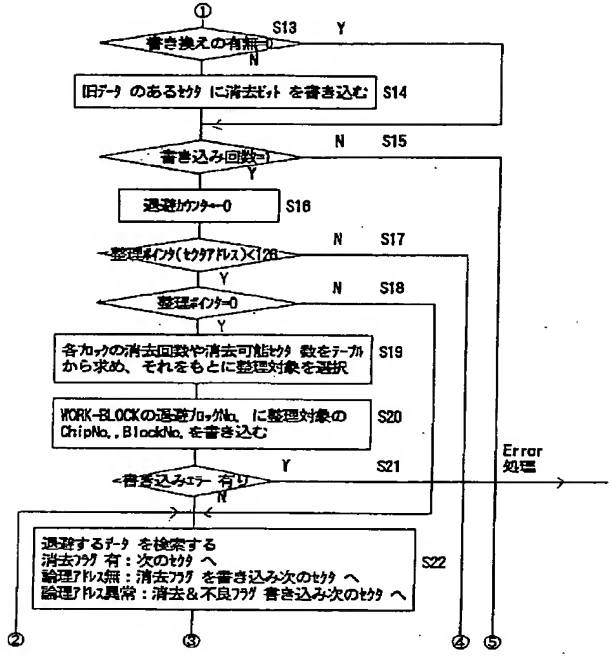
【図9】

記憶装置における書き込み時の処理を示すフローチャート



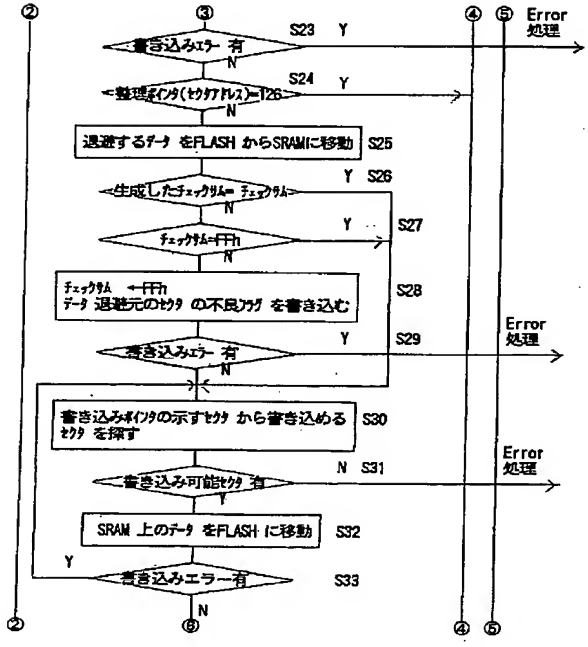
【図10】

記憶装置における書き込み時の処理を示すフローチャート（続き）



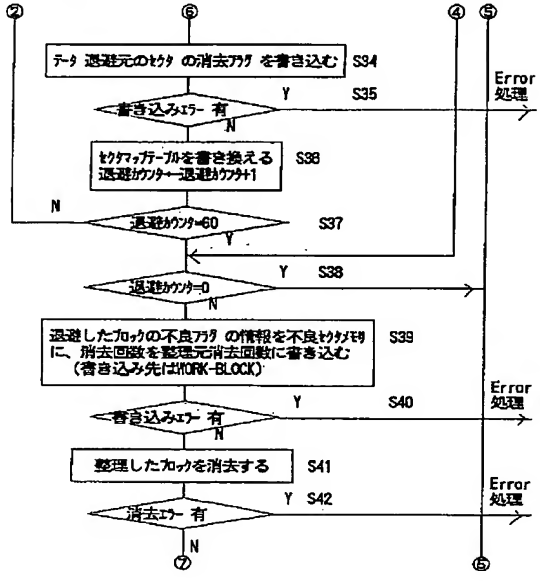
【図11】

記憶装置における書き込み時の処理を示すフローチャート（続き）



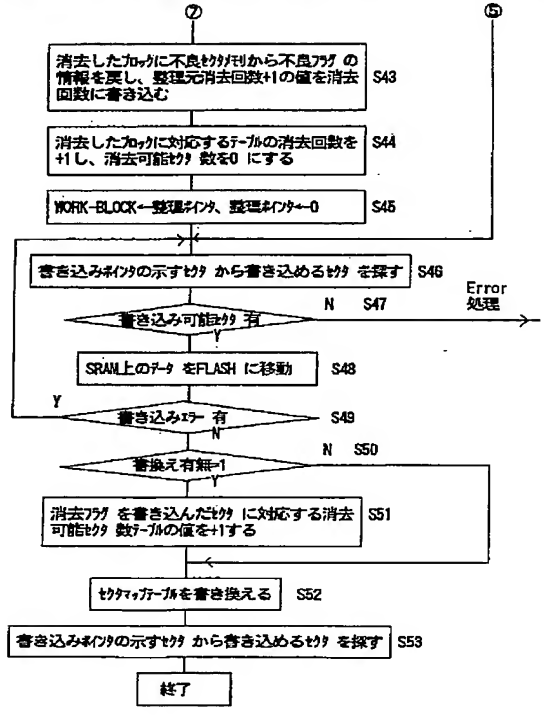
【図１２】

記憶装置における書き込み時の処理を示すフローチャート（続き）



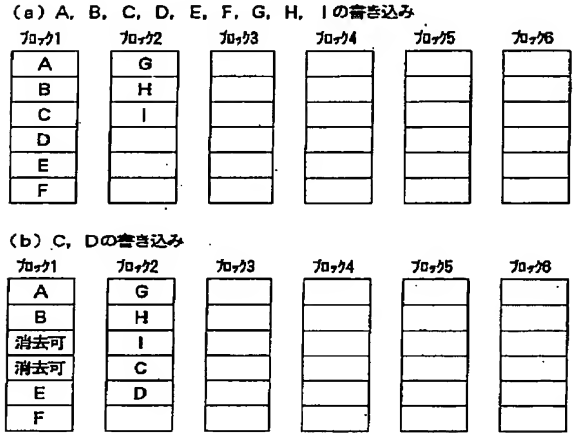
【図13】

記憶装置における書き込み時の処理を示すフローチャート（続き）



【図14】

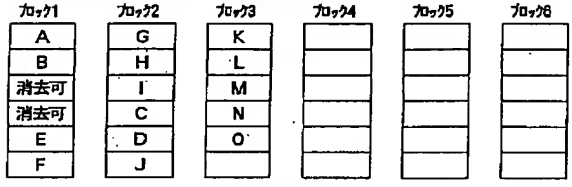
消去回数のバラツキを抑える実施例を示す図



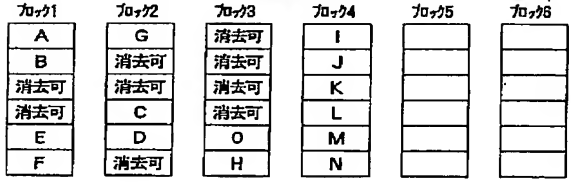
【図15】

消去回数のバラツキを抑える実施例を示す図（続き）

（c）J，K，L，M，N，Oの書き込み



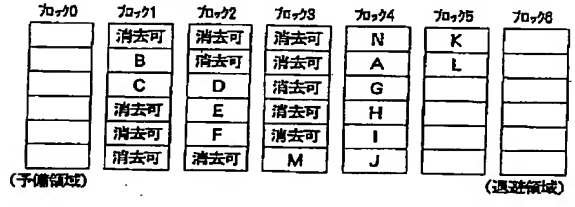
（d）H，I，J，K，L，M，Nの書き込み



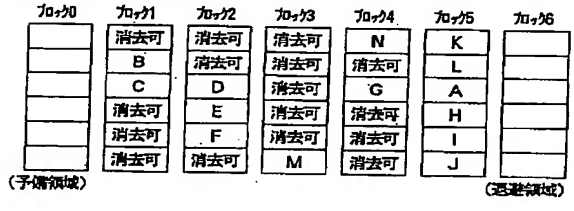
【図26】

空き領域の作成処理の実施例を示す図（続き）

（d）A，G～Lの書き込み

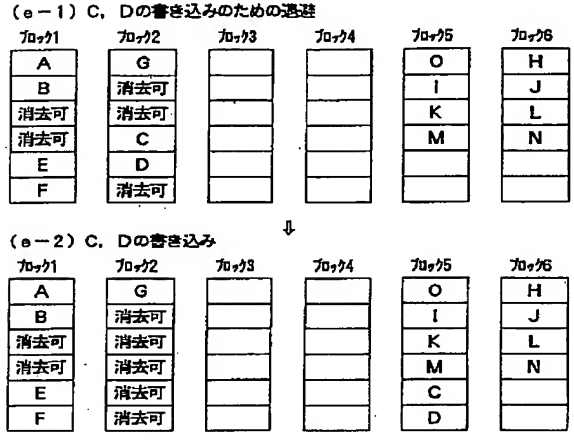


（e）A，H～Jの書き込み



【図16】

消去回数のバラツキを抑える実施例を示す図（続き）



【図17】

消去回数のバラツキを抑える実施例を示す図（続き）

（f−1）H，I，Jの書き込み（J書き込み時点で空きブロックなし）

ブロック1	ブロック2	ブロック3	ブロック4	ブロック5	ブロック6
A	G			O	消去可
B	消去可			消去可	J
消去可	消去可			K	L
消去可	消去可			M	N
E	消去可			C	H
F	消去可			D	I

⇩

（f−2）退避

ブロック1	ブロック2	ブロック3	ブロック4	ブロック5	ブロック6
A		G	J	O	
B		L	N	消去可	
消去可		H	I	K	
消去可				M	
E				C	
F				D	

⇩

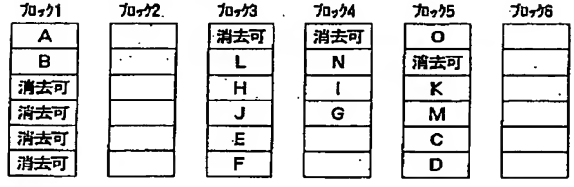
（f−3）Jの書き込み

ブロック1	ブロック2	ブロック3	ブロック4	ブロック5	ブロック6
A		G	消去可	O	
B		L	N	消去可	
消去可		H	I	K	
消去可		J		M	
E				C	
F				D	

【図18】

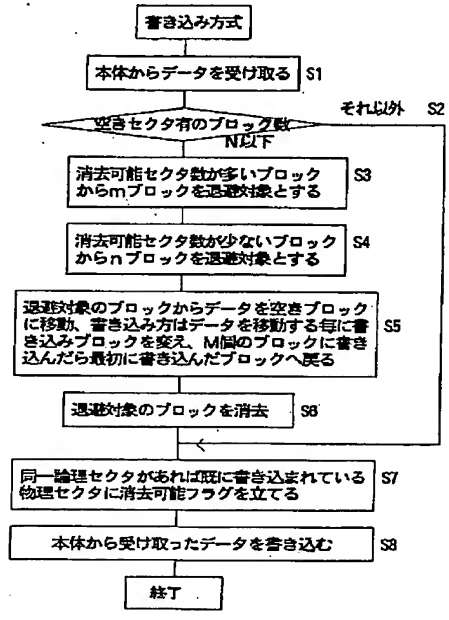
消去回数のバラツキを抑える実施例を示す図（続き）

（g）E，F，Gの書き込み



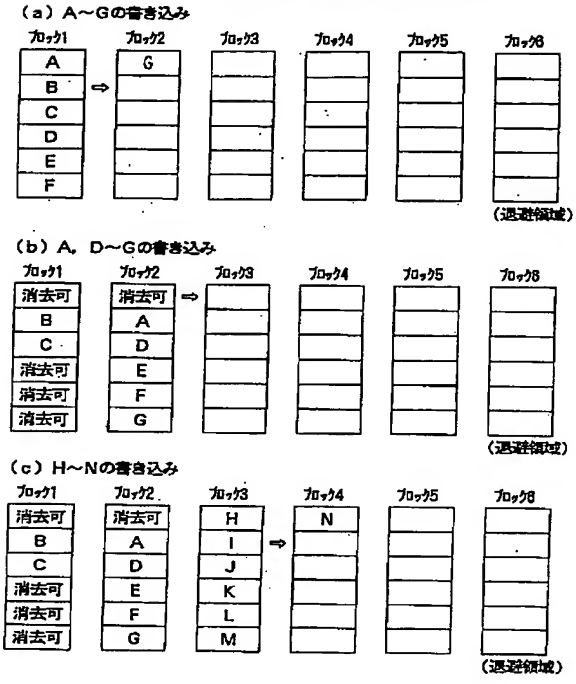
【図19】

消去回数のバラツキを抑える実施例の処理を示すフローチャート



【図20】

消去可能データをなくす処理の実施例を示す図



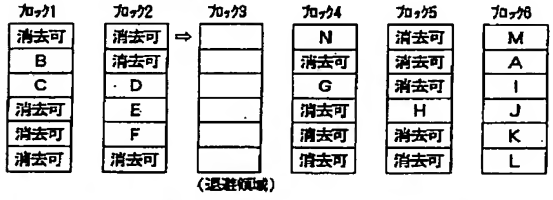
【図21】

消去可能データをなくす処理の実施例を示す図（続き）

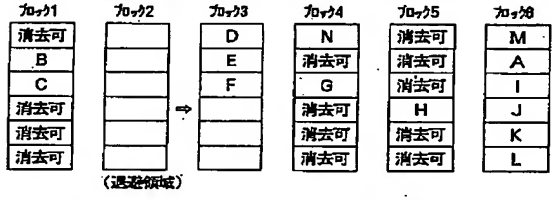
【図22】

消去可能データをなくす処理の実施例を示す図（続き）

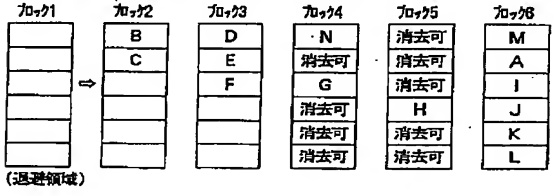
（f－2）A、I～Lの書き込み



（g－1）領域処理を行う（ブロック2とブロック3で退避動作）

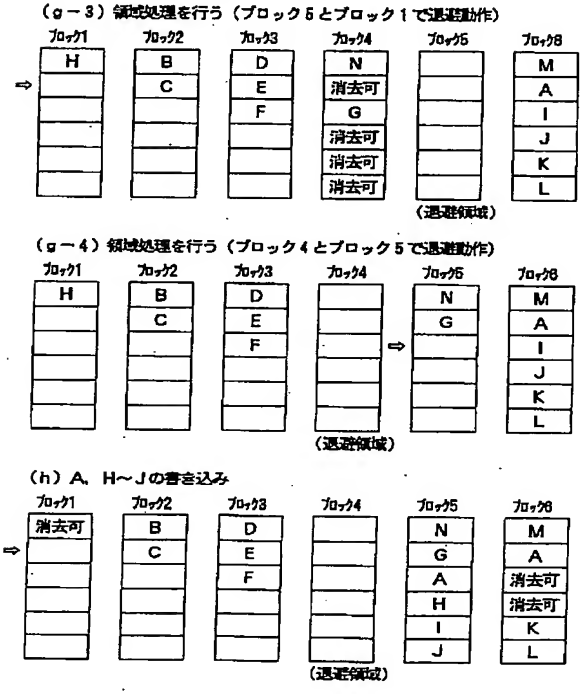


（g－2）領域処理を行う（ブロック1とブロック2で退避動作）



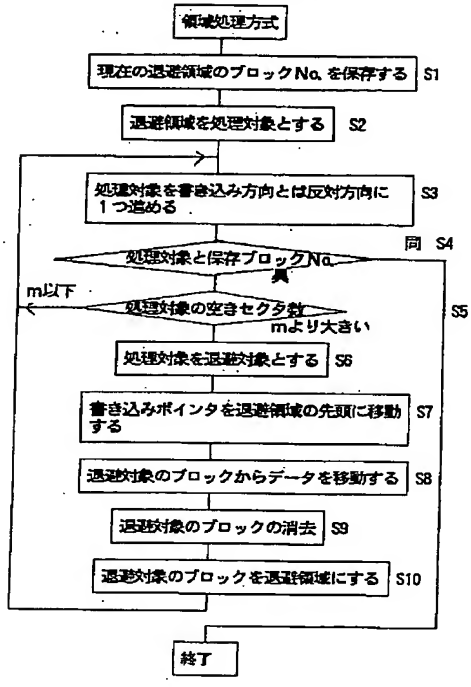
【図23】

消去可能データをなくす処理の実施例を示す図（続き）



【図24】

消去可能データをなくす処理の実施例を示すフローチャート



【図25】

空き領域の作成処理の実施例を示す図

【図27】

空き領域の作成処理の実施例を示す図（続き）

【図28】

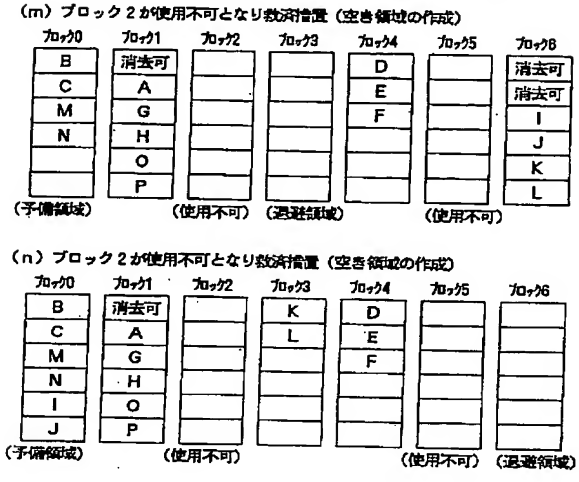
空き領域の作成処理の実施例を示す図（続き）

【図29】

空き領域の作成処理の実施例を示す図（続き）

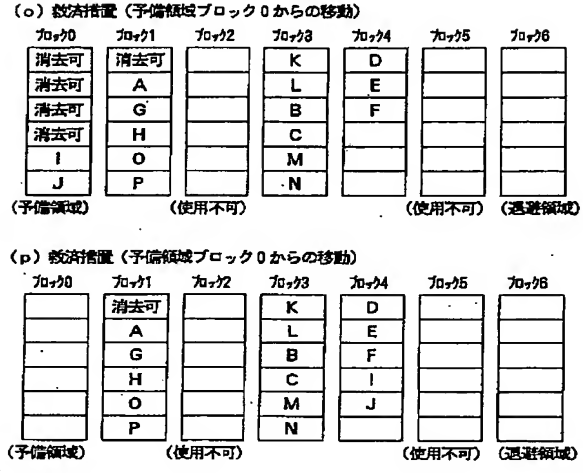
【図３０】

空き領域の作成処理の実施例を示す図（続き）



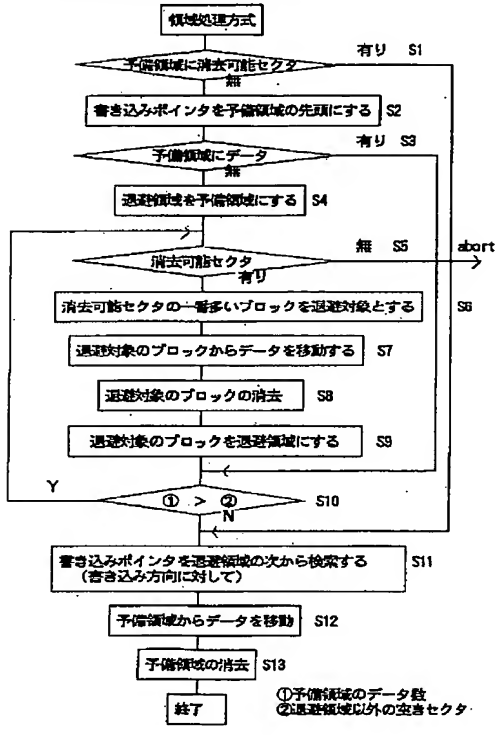
【図31】

空き領域の作成処理の実施例を示す図（続き）



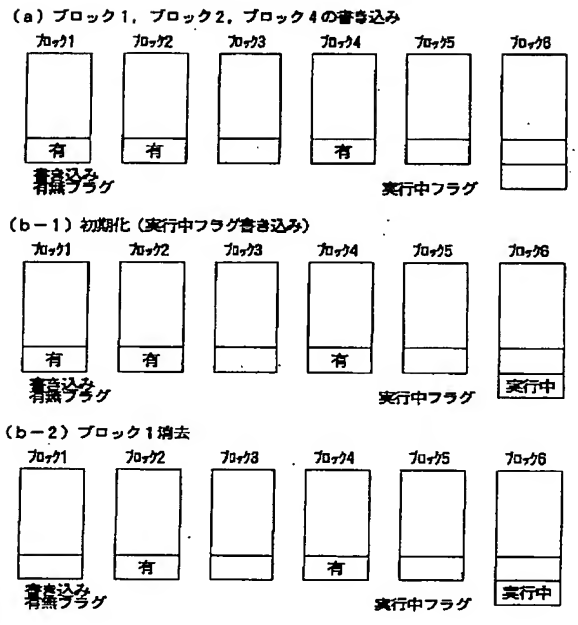
【図32】

空き領域の作成処理の実施例のフローチャート



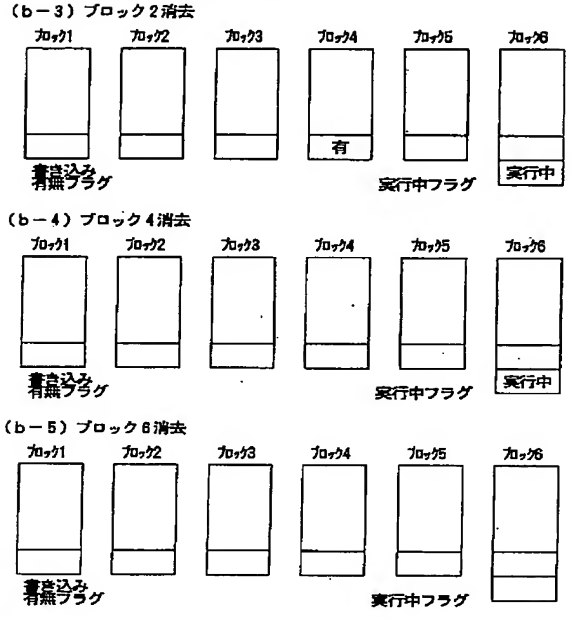
【図３３】

消去処理の実施例を示す図



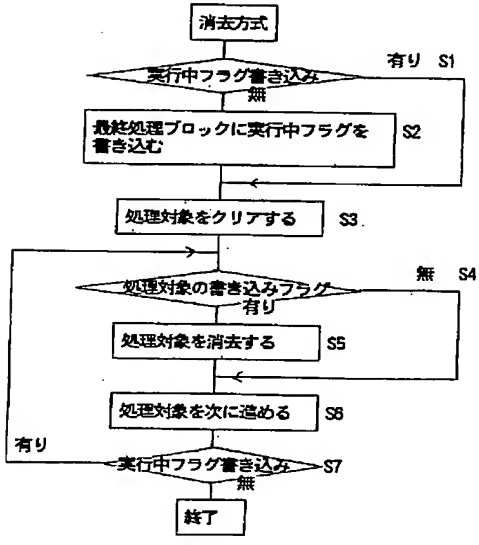
【図34】

消去処理の実施例を示す図（続き）



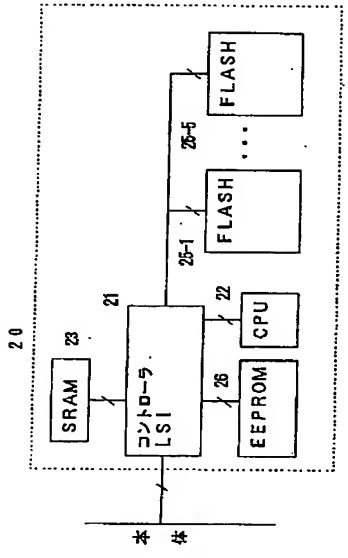
【図35】

消去処理の実施例のフローチャート



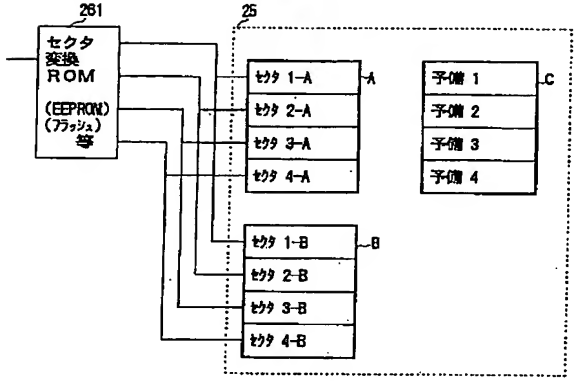
【図36】

書き込み速度の向上を図る実施例のシステム構成を示す図



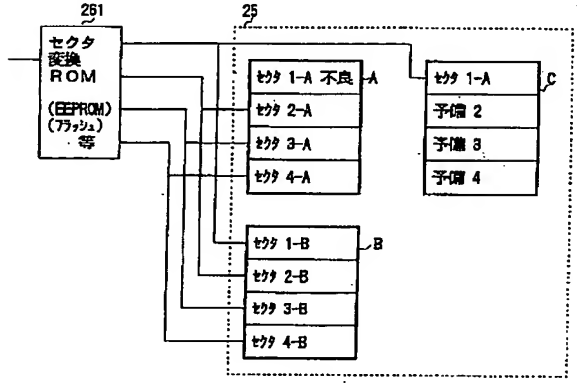
【図37】

書き込み速度の向上を図る第1の実施例を示す図



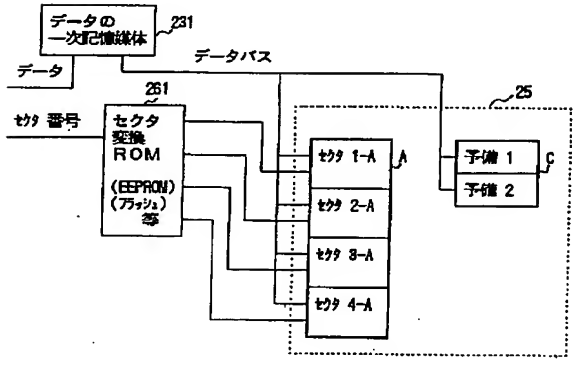
【図38】

書き込み速度の向上を図る第1の実施例を示す図（続き）



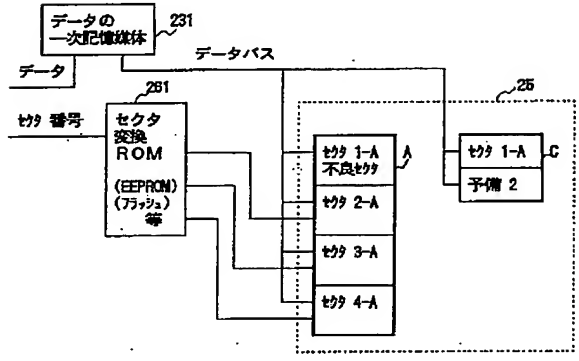
【図39】

書き込み速度の向上を図る第２の実施例を示す図



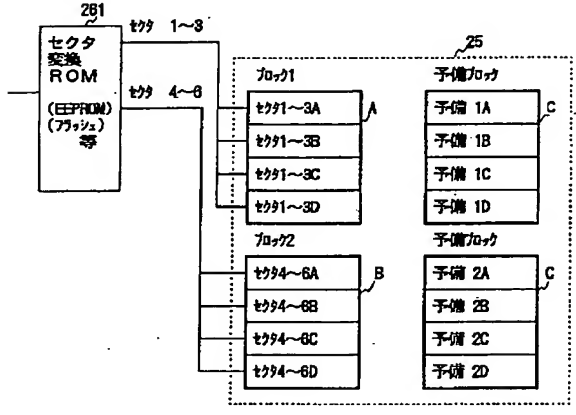
【図40】

書き込み速度の向上を図る第２の実施例を示す図（続き）



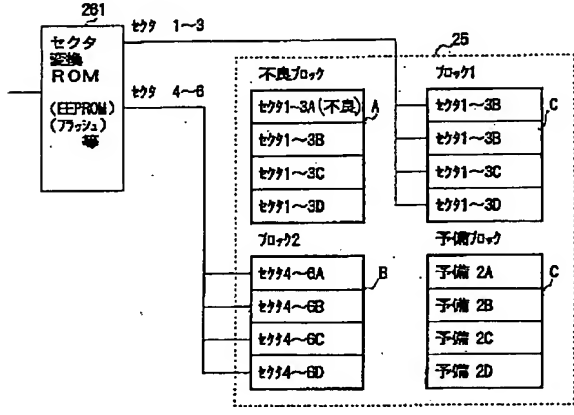
【図41】

書き込み速度の向上を図る第３の実施例を示す図



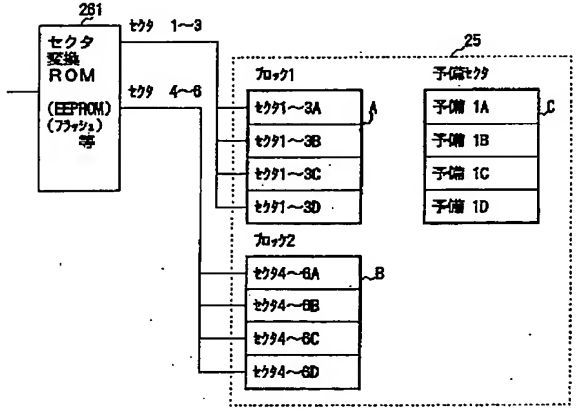
【図42】

書き込み速度の向上を図る第３の実施例を示す図（続き）



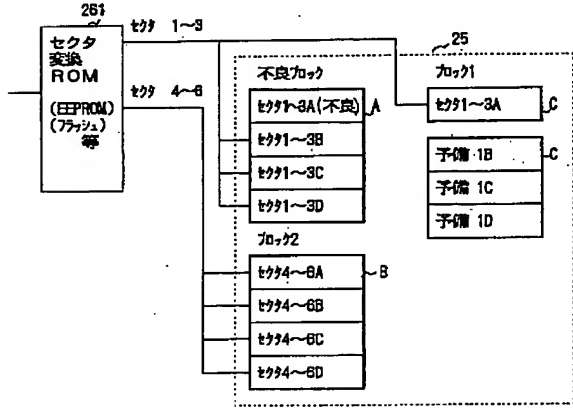
【図４３】

書き込み速度の向上を図る第４の実施例を示す図



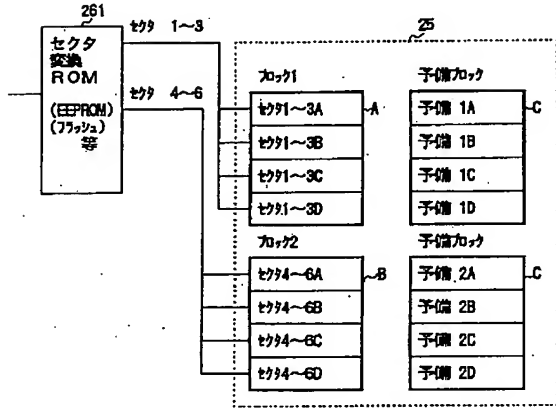
【図44】

書き込み速度の向上を図る第4の実施例を示す図（続き）



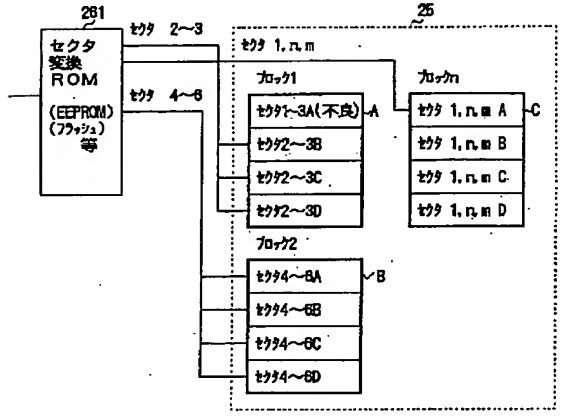
【図45】

書き込み速度の向上を図る第5の実施例を示す図



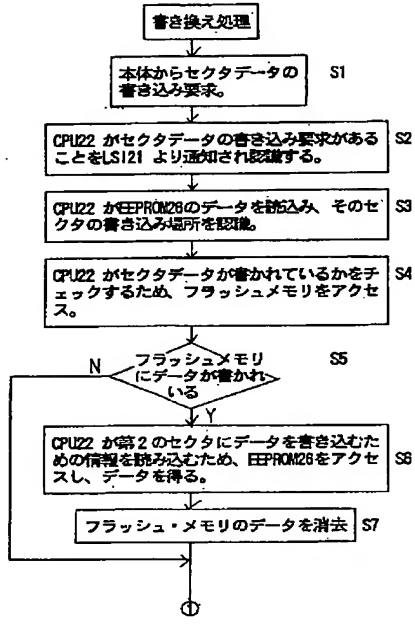
【図46】

書き込み速度の向上を図る第5の実施例を示す図（続き）



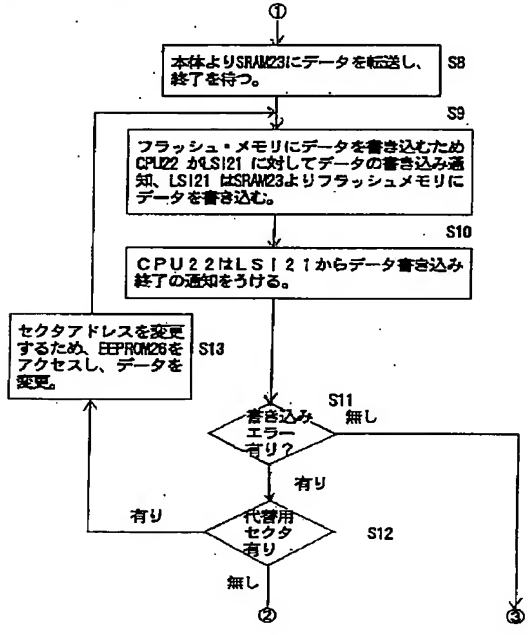
【図47】

書き込み速度の向上を図る実施例のフローチャート



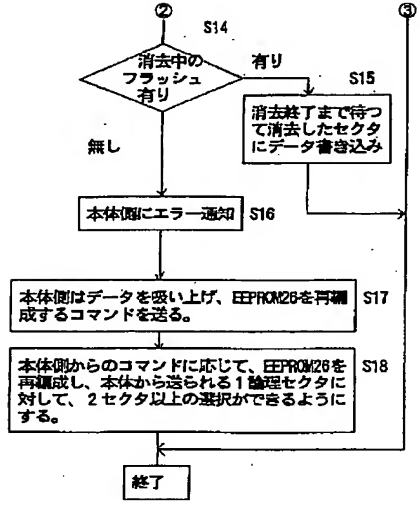
【図４８】

書き込み速度の向上を図る実施例のフローチャート（続き）



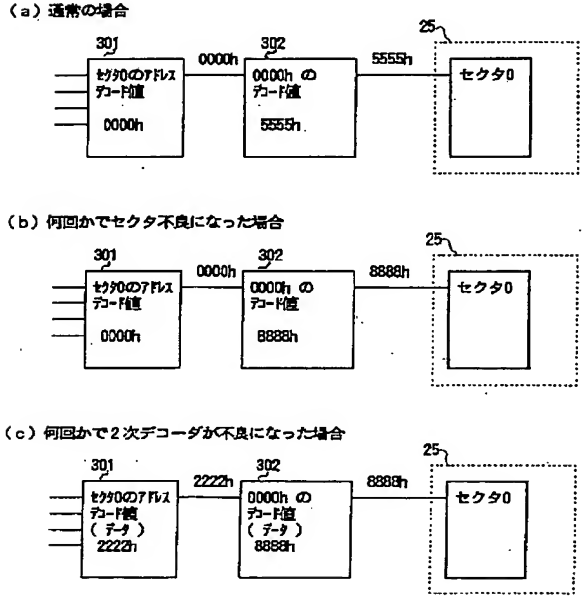
【図４９】

書き込み速度の向上を図る実施例のフローチャート（続き）



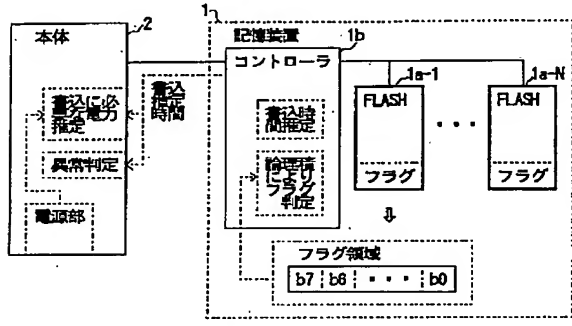
【図50】

アドレス変換用デコード部を2段とした実施例を示す図



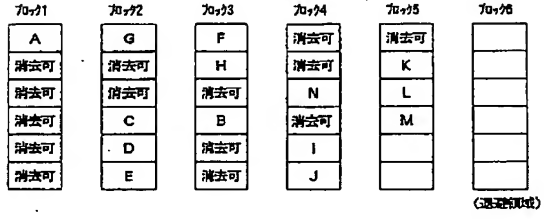
【図51】

書き込み時間推定処理を行うシステムの全体の概略構成を示す図



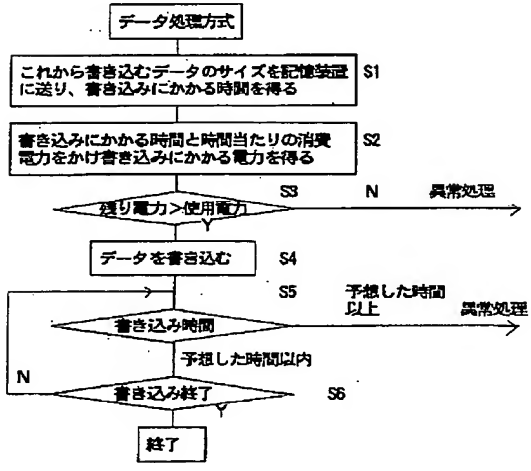
【図52】

書き込み時間推定処理の実施例における記憶領域の状態を示す図



【図53】

書き込み時間推定処理の実施例のフローチャート



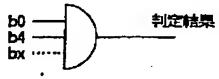
【図54】

フラグ判定処理の第1の実施例を示す図

(a) フラグレジスタ

b7	b6	b5	b4	b3	b2	b1	b0

(b) フラグ判定方法



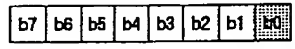
(c) 真理値表

行番	b4	b0	結果	
1	0	0	0	
2	0	1	0	⇐ (b0ビットが1固定不良となった場合)
3	1	0	0	⇐ (b4ビットが1固定不良となった場合)
4	1	1	1	

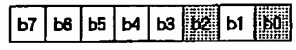
【図55】

フラグ判定処理の第2の実施例を示す図

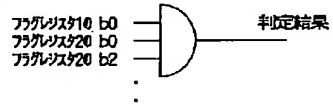
(a) フラグレジスタ1



(b) フラグレジスタ2

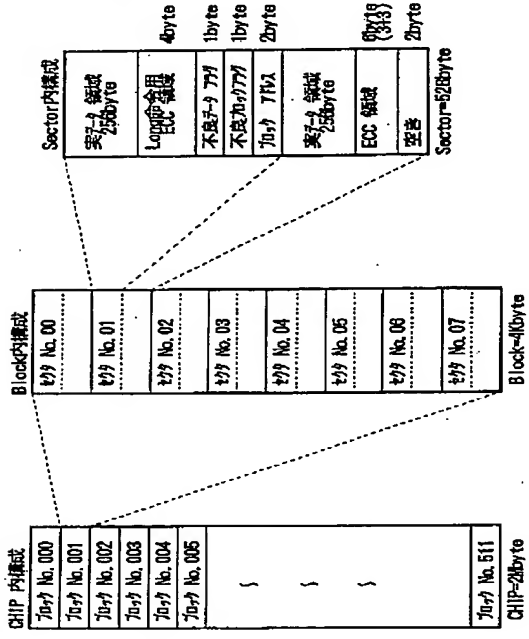


(c) 判定方法



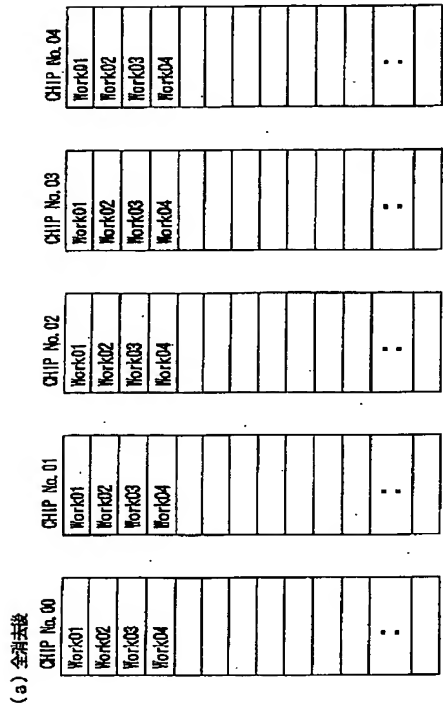
【図56】

管理テーブルを節約する実施例における記憶領域の構成を示す図



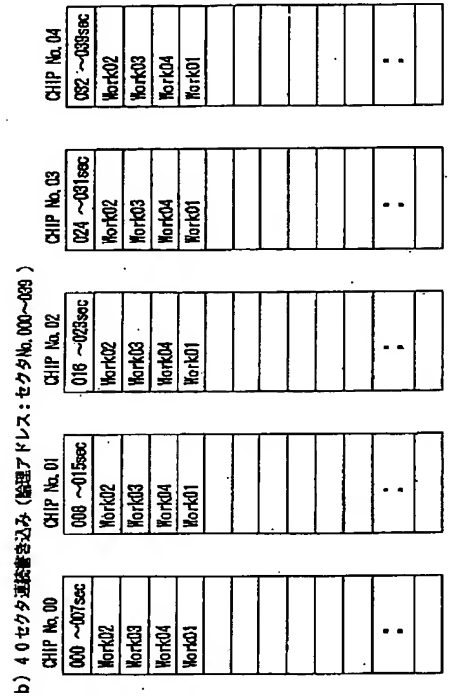
【図57】

管理テーブルを節約する実施例を示す図



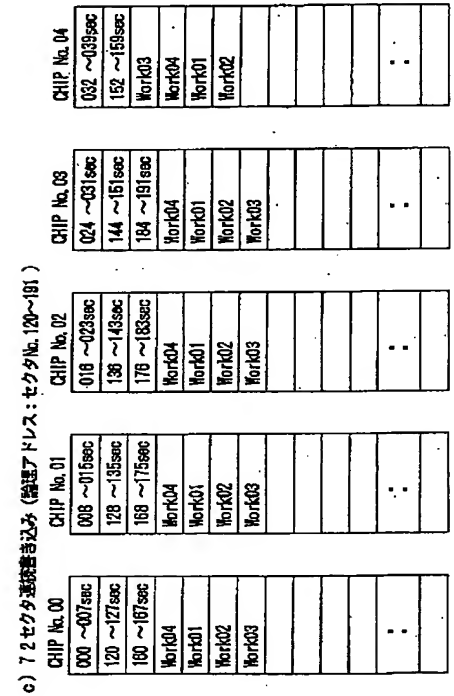
【図58】

管理テーブルを節約する実施例を示す図（続き）



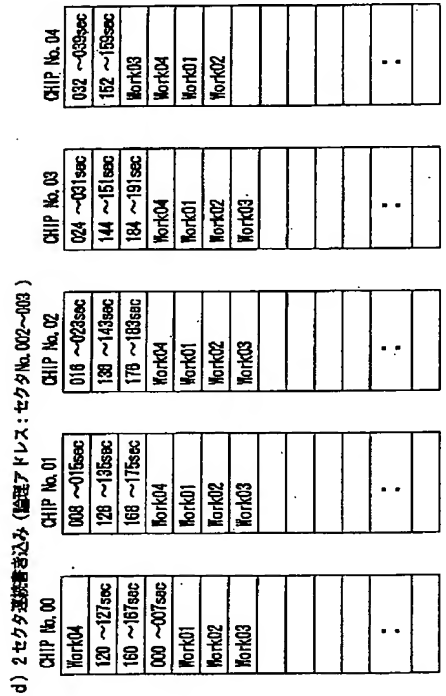
【図59】

管理テーブルを節約する実施例を示す図（続き）



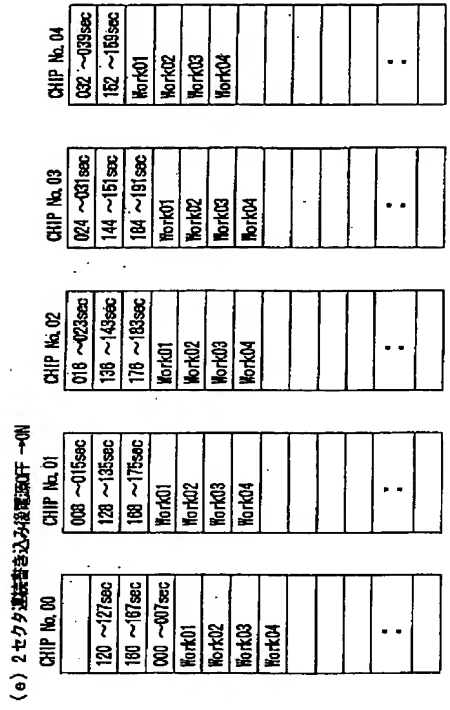
【図60】

管理テーブルを節約する実施例を示す図（続き）



【図61】

管理テーブルを節約する実施例を示す図（続き）



フロントページの続き

(51)Int.Cl.[7]	識別記号	FI		テーマコード（参考）
G11C 29/00	601	G11C	29/00	671B
	631		17/00	601C
	671			601B
				633A
				639B
				639C

(72)発明者 林 朋弘
神奈川県横浜市中区本町4丁目36番地 株式会社富士通コンピュータテクノロジ内
(72)発明者 柴崎 省吾
神奈川県横浜市中区本町4丁目36番地 株式会社富士通コンピュータテクノロジ内
(72)発明者 伊藤 裕之
神奈川県横浜市中区本町4丁目36番地 株式会社富士通コンピュータテクノロジ内
(72)発明者 竹原 勝
神奈川県横浜市中区本町4丁目36番地 株式会社富士通コンピュータテクノロジ内